# Rationale for developing the ALGAE Protocol

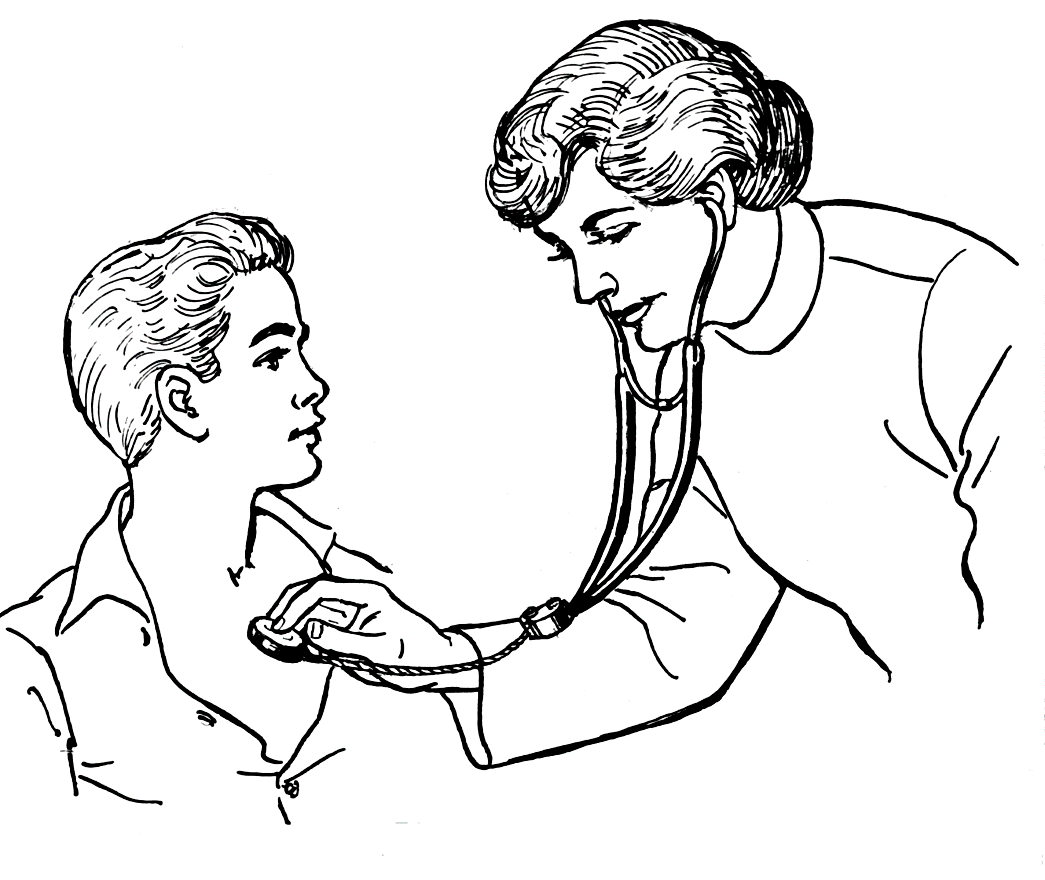
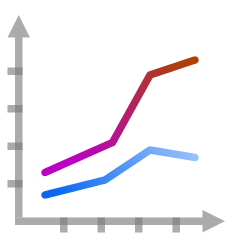
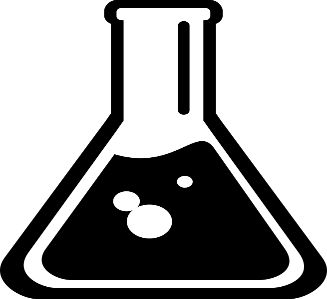
# 1. Introduction

The design of the ALGAE Protocol is based on requirements and assumptions that come from four distinct areas, listed in descending order of their priority for development:

1. **domain science decisions**: relate to the needs of satisfying the scientific use case
2. **business decisions**: constraints that govern how development needs to be done and how sensitive cohort data need to be handled
3. **data science decisions**: relate to how the protocol can provide value in the knowledge it produces
4. **software engineering decisions**: relate to making a solution that is well tested and performs well when it runs

The protocol was first shaped by the requirements of the air pollution study which funded its initial development. The implementation of the protocol was then greatly influenced by constraints relating to how the work of doing the protocol activity had to be done. Once we outlined a way of satisfying the domain science requirements within the context of business constraints, we turned our focus to providing features that added value to the basic results that had to be generated. During this period of data discovery, we learned more about the nature and quality of some of our data sets. Finally, software engineering concerns influenced aspects of testing and ensuring that the code could work well within a limited computing environment.

**2. Domain Science Decisions**



Exposure Science, Epidemiology and Research Publication Space

Our discussion about how the ALGAE Protocol was developed begins with requirements that come from the fields of epidemiology and exposure science which contribute to environmental health studies. The requirements reflect the interests of the researchers and the areas of contribution they want to make in a publication space. For now, we will focus on the requirements that need to be met to support Imperial College’s investigation of early life exposure to air pollutants and later life respiratory health in study members drawn from the ALSPAC cohort.

## 2.1 The Use Case Study

There is growing interest in the field of environmental health to examine the relationship between exposure to air pollutants in rapid stages of human development and health outcomes experience later in life. In particular, there is special interest in assessing exposures during pregnancy trimesters, when the rapid development of biological systems may make people particularly vulnerable to the effects of pollutants. The first requirement becomes:

|  |
| --- |
| **Domain Science-1**: The protocol will use residential address histories, historically modelled pollutant concentrations at residential addresses and information about life stages to assess life stage exposures in members the ALSPAC cohort. The life stage exposures will then be linked with health outcomes relating to respiratory health outcomes that are measured at years 8 and 15. |

## 2.2 Supporting Life Stage Calculations

The protocol needs to support two kinds of analysis, each of which will examine different life stages. In the use case study, the early life analysis uses pregnancy trimesters and the first year of life. In a later life analysis, the life stages include years of life that bridge the time between when the early life analysis ends and when the health outcomes taken at year 15 occur.

|  |
| --- |
| **Domain Science-2**: The protocol will support both an early life and a later life analysis. The early life analysis will assess exposures in: Trimester 1 (T1), Trimester 2 (T2), Trimester 3 (T3), and an early life period that spans the first year of life (EL). The later life analysis will assess exposures for years of life beginning in year 1 and ending in year 15 (YR1...YR15). |

In order to make the assessments, we need to identify the start and end date of each life stage. Calculations for life stages in the early life analysis depend on knowing the conception date of study members:

|  |
| --- |
| **Domain Science-3**: Date of Conception will be calculated as: date of birth – (7 x gestation age at birth measured in weeks) – 1 day. |

Given the conception date, the calculations for early life stages become:

|  |
| --- |
| **Domain Science-4**: The temporal boundaries of early life stages will be calculated as follows: T1 = [conception date, conception date + 92 days], T2 = [conception date + 93 days, conception date + 183 days], T3=[conception date + 184 days, birth date – 1 day], EL=[birth date, birth date + 1 year – 1 day]. |

This way of calculating boundaries will work for most study members, but will generate errors in cases of very premature births. For example, suppose a study member has a gestation age at birth of 25 weeks. This represents a period of 175 days, which means that the boundaries for both T2 and T3 will overlap with the EL period. In order to prevent exposure days from being double counted in life stages, we will need to support a feature for making any corrections:

|  |
| --- |
| **Domain Science-5**: In the early life analysis, the protocol will need a facility for correcting overlaps in life stage boundaries that may occur for prematurely born cohort members. |

The later life analysis requires only a birth date:

|  |
| --- |
| **Domain Science-6**: The boundaries of life stages in the later life analysis will be calculated as follows: YRn = [date of birth + n years, date of birth + (n+1) years – 1 day. |

## 2.3 Supporting Assessment of Specific Pollutants

The use case study was funded to examine the effects of specific types of air pollutants, which could be generated from different sources:

* PM10\_tot: PM10 particulate matter coming from all sources
* PM10\_rd: PM10 particulate matter coming from roads
* PM10\_gr: PM10 particulate matter coming from sources other than roads
* NOx\_rd: nitrogen dioxide pollution coming from roads.
* NAME: high altitude pollution that comes from outside the exposure area.

When we discuss business decisions, it will become clear that there is good motivation to make the protocol independent of any specific pollutant. However, for now, we will acknowledge the specific needs of one use case study:

|  |
| --- |
| **Domain Science-7**: The protocol will assess exposures for various pollutants. |

## 2.4 Supporting Different Spatial and Temporal Resolutions for Exposure Data

The temporal and spatial resolutions of exposure values are important features that can help distinguish the publication niches of exposure studies. Examples of temporal resolutions include hourly, daily, monthly and yearly values, whereas examples of spatial resolutions include region, district, output area, super output area and residential address.

Another important aspect of assessing exposures is the way sample exposure values are treated as proxies to make more general assumptions about a broader exposure assessment. For example, the daily exposure on January 1 could be used as a proxy to represent the exposure on a typical day of a calendar year. As another example, a postal code could be used to represent the locations of study members who may have moved only a couple of houses away from their last addresses.

The resolution and sampling of exposure values can be influenced by a number of factors such as:

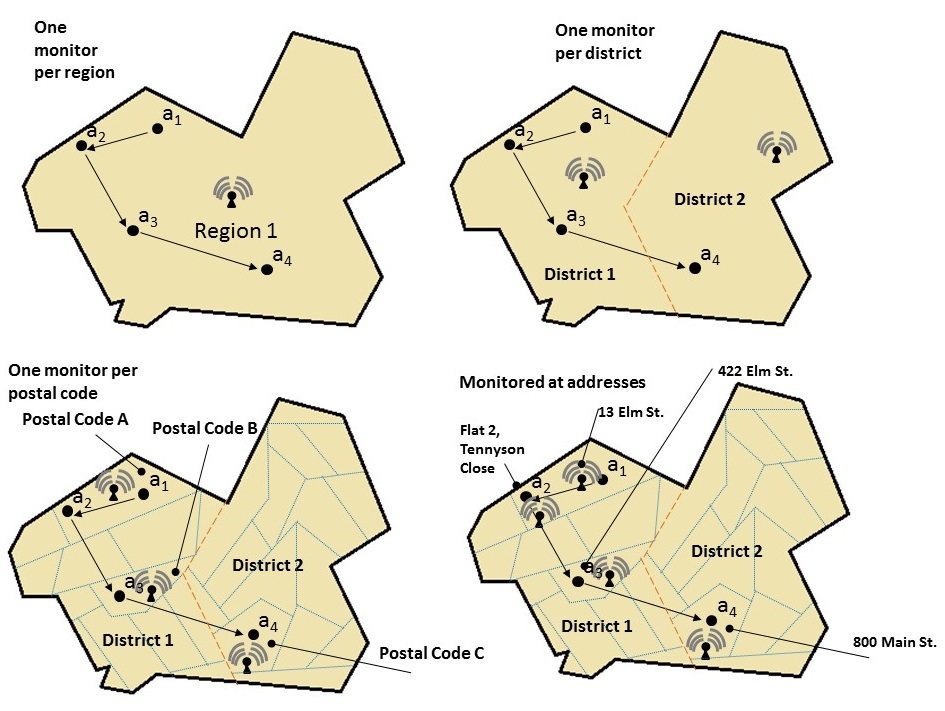
* the availability of mobility data
* whether the researchers believe that an exposure area may have exhibited high levels of spatial and temporal variation during an exposure time frame
* the computational resources needed to generate the exposures

In many environmental health studies, detailed mobility data about study members can be difficult to obtain. In many early life exposure studies, researchers use the birth addresses of study members to represent their location for their entire gestation period. One reason why this is a popular option for researchers is that the birth address is often found on birth records. It can be challenging to obtain a more detailed residential address history from the mothers, especially in cohorts where members may have been born decades ago.

Study members may move around in exposure area, thereby raising the question of whether, for a given period, their location at one place or another can make a significant difference to their exposure assessment. For example, it is well known that between 20% and 30% of women move when they are pregnant. If one study used the address of study members at birth, and another study considered all the addresses their mothers occupied during pregnancy, would the exposure assessments be significantly different?

Some important studies have concluded that birth addresses are an accurate proxy for representing the complete residential address history that may have been used during pregnancy and some of the post-natal period. However, the authors tend to acknowledge that the conclusion can depend on aspects of temporal and spatial resolution used to assess exposures.

To show an example of how sampling of exposure values can affect perceptions of variability in exposure, consider the following exaggerated scenario. Suppose study member 1034A moves 4 times and multiple studies having varying geographical resolution attempt to assess 1034A’s exposure to air pollution.



*Fig. 1: Illustrating the effect of sampling on perceptions of whether moving makes a difference in exposure assessments*.

In the study which uses one monitor to represent the exposure for an entire region, the interpolated differences in exposures amongst locations a1, a2, a3 and a4 may not appear significant. When one monitor per district is used, the estimates may show more variation because the input of more pollution monitors is being considered.

The example oversimplifies the process of exposure assessment, but it is meant to convey the idea that exposures may vary considerably depending on what temporal and spatial resolution we consider. Underlying our perception of exposure in the assumption about whether an exposure area will exhibit high spatial and temporal resolution.

In the use case study, the exposure scientists anticipated that most members of the cohort may have been gestating during years that could have been marked by a period of deindustrialisation. They also thought the pollution may have varied significantly between rural and urban areas. Note that they did not know *a priori* whether their predictions would be supported by results.

Had they used low spatial and temporal resolutions, the exposure scientists could have significantly reduced project costs by generating fewer exposure values. However, the main driver for supporting a high resolution exposure study came from the project’s epidemiologists. They thought that periods of rapid development may have made study members susceptible to being affected by historical exposures.

The short duration of these life stages could produce potentially very small values for measurements such as cumulative exposure. The variation in small exposure values could be an important factor in allowing scientists to observe trends in the health outcome studies that rely on the exposure values. Therefore, in order to best capture any variation in small values, the scientists wanted the protocol to use high resolution exposure values.

Whereas short life stages featuring rapid development may warrant using high resolution exposures, longer life stages featuring slower development may warrant using low resolution exposures. For example, study members will evolve physiological systems in Trimester 1 at a faster rate than they would for their entire 8th year of life. If we follow the assumption that they would be less susceptible to pollution exposures during slower periods of development, then it may be sufficient to assess exposures using weighted annual exposures rather than daily exposures. There was a need to cater the resolution of exposures to the nature of the life stages being examined.

Our discussion about resolution in exposure values leads to the following design points:

|  |
| --- |
| **Domain Science-8**: Assume that the exposure will exhibit high spatial and temporal variation during the exposure time frame. |

|  |
| --- |
| **Domain Science-9**: Assume that the susceptibility of cohort members to the spatial and temporal exposures may depend on the life stages being used in assessment. |

|  |
| --- |
| **Domain Science-10**: The early life analysis will use daily exposure values and the later life analysis will use annual average exposure values. |

The ALGAE Protocol is designed to support an exposure study where there may be high levels of spatial and temporal resolution in the exposures. Rather than compelling researchers to simplify their assessments by using more generalised proxies, ALGAE provides researchers the ability to ignore or consider results that are derived from fine-grained precision values. Next, we must consider how that precision is preserved or lost through aggregation calculations.

## 2.5 The Need to Aggregate Exposures in Different Ways

Discussions with the scientists revealed that exposures would have to be aggregated in different ways. The choice of which aggregation to use for which part of their study depended on four factors:

* the nature of how the specific pollutants affect the human body
* the ability to assess the effect of extreme values found in the aggregated exposures
* the skill of the developer
* the ability for the software to support statistical operations.

A cumulative exposure may well suit various types of particulate pollutants which could accumulate in the lungs or other tissues over a long period of time. An average exposure may be more appropriate to use for gaseous pollutants which could accumulate in and then dissipate from the body over a short period. Median exposure would help statisticians assess the influence of extreme values on average or cumulative values.

The variety and complexity of aggregation operations that were included in the analyses were limited both by the skill of the developer and by the choice of technology used to support the protocol. Someone who is skilled as a software developer may not necessarily be actively skilled in advanced statistics. And yet the developer would need to be able to both articulate how a statistical operation would be done and know how to implement it using the technology that was chosen to support the protocol. As we will discuss later, we decided to use PostgreSQL to write the code used by ALGAE. However, PostgreSQL’s support for statistical operations is not as broad as that of statistical technologies such as SPSS, STATA or R.

The need to support multiple aggregation operations forms the next design principle:

|  |
| --- |
| **Domain Science-11**: The protocol will aggregate exposures by life-stage, and express the results using cumulative, average and median exposures. |

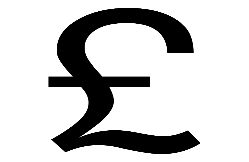
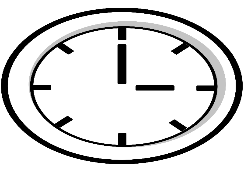
## 2.6 Summary

In this section, we have identified what the protocol must do in order to support the research interests of epidemiologists and exposure scientists, and to help them distinguish themselves in a timely research publication space.

The protocol will use residential address histories, historically modelled pollutant concentrations and life stage data to assess life stage exposures in study members drawn from a longitudinal cohort. Life stage assessments will be made for each of PM10\_tot, PM10\_rd, PM10\_gr, NOX and NAME pollutants.

ALGAE will support an early life analysis which aggregates daily exposure values to produce results measured across each pregnancy trimester (T1, T2, T3) and the first year of life (EL). The protocol will also support a later life analysis which uses aggregated annual exposures to produce results measured across years of life from year 2 to year 15 inclusively (YR2….YR15). Calculations for the boundaries of life stages need to be clearly defined and ensure that each day of an exposure time frame belongs to exactly one life stage. Exposures will be aggregated using cumulative, average and median values for each life stage for each pollutant for each study member.

**3. Business Decisions**



Time, money, multi-site logistics, security and reusable research assets

Whereas the previous section described what the protocol had to support, this section describes the constraints under which that support could be made. The most significant constraint which influenced protocol development was that in order to produce results based on cohort data, the protocol could only be run off-site within a secure, limited computing environment. In order to mitigate the cost of short yet expensive windows of off-site development, we had to find ways of doing most of the protocol development without real data. The constraints of off-site development led us to support a protocol which could be driven off a fake test data sets.

As Imperial College continued to expend more resources to complete its complex use case study, it began to consider the pros and cons of making a generic data set which could support multiple studies rather than just one. As interest in applying the protocol for different cohorts grew, we had to consider the costs, risks and benefits of trying to make our specific protocol more generic.

## 3.1 The Need to Support Information Governance Policies of Cohorts

ALGAE’s design had to respond to two concerns about handling sensitive, identifiable cohort data. First, the protocol had to be able to strip out any data from study members who withdrew their consent to participate in either the study or the cohort project in general. Second, the way it linked data sets together and the way it produced results had to minimise the identifiability of study members who remained in the study.

Supporting a study member’s right to withdraw from a study will become a common requirement for protocols used in environmental health studies. As we shall see later in the discussion about business decisions, activities such as the use case study can take years to set up, gather information, perform analyses and produce results. At any time during that activity, a study member may want to withdraw their permission to participate in the activity. Therefore, the protocol needs to be able to omit these people from all analysis.

|  |
| --- |
| **Business-1**: The protocol must be able to omit from analysis any cohort members who have withdrawn consent to participate in the activity, at any time during the activity. |

### 3.1.1 Supporting Constraints of Physical Security

For study members who remain, the protocol must support a work environment that minimises the risk of inappropriately disclosing sensitive data that could identify them. There are many ways to provide security safeguards in data handling, but we’ll begin with the physical aspects of how data are stored and processed.

When a research group collaborates with a longitudinal cohort study, the researchers must be formally approved by the cohort and they must sign forms to indicate that they will comply with the cohort’s requirements for handling and sharing data. The terms of the cohort’s access agreement should ensure that data are handled in an appropriate manner.

However, in practical terms, once data files physically leave a cohort’s facilities, what happens to them when they are managed off-site is beyond their control. No matter how vigilant collaborating researchers may be, there will always be a risk that the data could be accidentally disclosed.

In an effort to minimise this risk, there is a growing trend for cohorts to require that as much of a protocol as possible is applied within their own secure premises. This measure ensures that sensitive data remains on-site. A cohort’s information security staff can inspect the results and decide what data sets may be taken off-site.

Ideally, in order to minimise any risk of inappropriate disclosure, the cohort would use its own staff to generate and inspect the results. The external collaborators would then obtain results that were sufficiently anonymised to make identifying individual study members improbable. However, as is often the case in collaborations between research groups and cohorts, the cohort tends to provide the data sets and the research group tends to provide niche expertise for doing the analysis and producing results. The division of labour raises an issue about which study partner will carry out work and what physical location they will use to do so.

In the use case study, ALSPAC provided the wealth of data related to: the movement patterns of their study members; birth data to help establish early life stage boundaries; and health outcomes data from medical tests that related to respiratory health. Imperial College provided expertise in the areas of epidemiology and exposure science. The exposure scientists have great skill in handling geospatial data, and they were responsible for generating historical exposure values for addresses that were used by study members.

The birth data, health outcomes and residential address histories were too sensitive for the data sets to leave ALSPAC’s premises in Bristol. However, it was not feasible for Imperial College to use its own complex in-house systems to generate exposure data at ALSPAC’s premises. Therefore, the protocol had to support a work activity where parts of it would be done at the cohort’s facilities and parts of it would be done at the offices of the partnering research group.

At this point in our discussion about physical handling of sensitive data, it is useful to divide a protocol’s effort into three parts: input data, intermediate data sets and results. In the use case study, our main input data sets include:

* **residential address histories**: a chronology of address periods, each comprising a residential address, a start date and an end date
* **life stage data**: including birth dates, gestation age at birth to help establish life stage boundaries
* **exposure data**: historical exposure values generated for addresses taken from the residential address histories

The intermediate data sets link these data sets together, and the results assess exposure for each pollutant for each life stage for each study member.

ALSPAC’s own information governance policies mandated that because the study used birth dates and residential address histories, linking the cohort data sets and generating results could only be done physically onsite at its own secure facilities in Bristol. Life stage data and residential address histories would not be allowed off-site, but Imperial College would be allowed to use its own expertise to generate exposures for relevant addresses at their offices in London.

This forms the next two design influences:

|  |
| --- |
| **Business-2**: The intermediate data sets that are made from linking cohort data and the results produced by analysis can only be created on-site at the cohort’s premises, using a secure and limited computing environment. |

|  |
| --- |
| **Business-3**: Scientists could only take aggregated exposure results off-site. They would not be allowed to take away results which contained birth dates or which linked residential addresses to specific individuals. |

|  |
| --- |
| **Business-4**: The protocol assumes that exposure data is generated on the premises of the collaborating research institution and not those of the cohort facilities. |

### 3.1.2 Minimising the Identifiability of Generated Exposure Values

Imperial College was faced with the challenge of developing a protocol where almost all the data and linking activities would have to be done off-site in Bristol. The exception was that exposure scientists had to be able to generate historical exposure values that could be generated in London and linked with other data sets in Bristol.

In order to generate exposures, the Imperial exposure scientists would need to match residential addresses with map coordinates. These sets of coordinates would then be provide inputs to exposure modelling software which would use historical pollution data to produce concentrations of pollutants for specific addresses for specific times.

In order to minimise the identifiability of study members, Imperial College was given the following data from ALSPAC:

* all residential addresses ever occupied by any cohort member, whether or not the addresses could be used in the early life analysis, the later life analysis or neither
* the earliest conception date of any study member
* the latest date of any study member which represented the last day of their first year of life
* the set of calendar years which would span the first fifteen years of life of all study members

The concern about minimising identifiability of residential addresses that were taken off-site from ALSPAC compelled Imperial College to use a brute force approach to generating vast amounts of exposure data. The exposure scientists had all the residential addresses, but they could not tell which study member occupied which address and when. The intentional introduction of these unknowns meant that Imperial’s exposure scientists had to assume that each residential address could have an equal likelihood of being used by any study member at any time in his or her early life or later life time frames.

For the early life analysis, Imperial generated daily exposure values for each pollutant for each location for each day of the overall early life time frame. The time frame began with the earliest conception date and ended with the latest date for the last day of first year of life. In the later life analysis, those same addresses were used to produce annual exposure values for each pollutant for each year of the overall later life time frame. That time frame began with the earliest first year birthday and ended with the latest last day of the fifteenth year of life.

The exposure modelling software took months of continuous operation to produce the data set. During a power outage at Imperial College, the process had to be restarted. Much of the generated data would be ignored when individual exposure records were matched by location to fit the start and end dates of residential address history records. Thus, in order to minimise the identifiability of study members, the exposure scientists generated far more exposure data than the results would actually require. The sheer volume of data would bear part of the business cost of doing off-site visits.

The task of minimising identifiability of exposure data influences the design of the exposure data sets as follows:

|  |
| --- |
| **Business-5**: Assume that exposure data for both early and later life analyses will be generated based on a set of all residential addresses that have ever been occupied by any study member for any period that falls within either their early life or later life analysis. |

|  |
| --- |
| **Business-6**: An exposure record will comprise a geocode, a date of year, and concentration values for each pollutant. In the early life analysis, the date of year will represent a daily exposure record. In the later life analysis, the same field will represent an annual exposure value and will be January 1st of some year that is involved in the later life analysis. |

## 3.2 The Need for a Fully Automated Protocol

Until now in our discussion, we have said nothing about whether the protocol would be implemented using manual steps, automated steps, or a combination of the two. The decision was driven mainly by the need for the Imperial College team to minimise the potentially prohibitive cost of developing the protocol off-site in Bristol. Once it was clear that the developer would have to take exposure data generated in London and link it in Bristol, the next business question was: “How long will it take to generate results?”

At least four factors made it difficult to estimate the time needed to obtain results:

* the volume of exposure data that would have to be linked with other data sets
* the availability of high performance computing resources to support applying the protocol
* the work needed to clean and repurpose residential address history data
* the difficulty for the developer or domain experts to determine whether the results were correct

As we have already discussed, there was a need to link a vast number of exposure records with residential address histories. It was obvious that this activity could take a considerable amount of time to complete, but how much time would depend on the computing resources that were available when it was run.

The availability of fast or slower machines was dependent on how ALSPAC prioritised computing resources for various visiting researchers and for the needs of its own staff. Even if Imperial College could have been able to run the protocol on cohort data at its offices in London, its estimates for performance would not likely apply to the target computing environment in Bristol.

As we will discuss in the data science decisions, it became clear that the nature and quality of the residential address histories had not been explored by any of the scientists either at Imperial College of the University of Bristol. It was therefore not known what errors we would encounter or how relevant any anomalies we observed in the data would be to the results.

Finally, the constraints about where data could be linked and what could be taken off-site from ALSPAC’s facilities made it challenging for Imperial College’s team to determine whether the protocol would produce correct results. The developer, located on-site in Bristol, had access to all of the data but lacked enough domain knowledge about exposure science to determine whether results were sensible. The exposure scientists, located back in London, were able to identify anomalous aggregated exposures. However, they were not able to suggest a cause because they didn’t have the sensitive input data available that would let them trace results.

Given the difficulty of estimating beforehand how long it would take to apply the protocol, Imperial College decided to schedule visits to Bristol in week-long blocks. Each visit then came with four major costs:

* the transportation costs to visit Bristol
* the costs of maintaining a developer off-site in a hotel
* the personal cost to the developer for working away from home for prolonged periods
* cost of coordinating people resources at both institutions

The short, expensive windows of development could be easily taken up with any of the factors that made it difficult to estimate the time to apply the protocol. Moreover, if there were errors in the results, a follow-up visit would be required that would involve applying the protocol again. There was the further cost that any refinements to features of the protocol would also require another visit.

Returning to our decision about the extent of automation that was needed in the protocol, we decided that it needed to be a single program that could run from start to finish without manual involvement. We did not have the resources to afford any form of manual linking or cleaning, let alone the resources to do it again. A fully automated protocol also had the benefit of being able to run overnight when the developer was sleeping!

Our next decision becomes clear:

|  |
| --- |
| **Business-7**: The protocol must be fully automated. |

## 3.3 The Need to Support Testability

In order to minimise the need for return visits to Bristol and to allow as much of the protocol to be developed at Imperial College as possible, we had to ensure that the software we created could be tested. A set of fake data sets would be developed which would demonstrate that various program features were working correctly. Once the program passed a collection of test cases, then presumably the cost of doing test-and-fix with real data would be minimised. It is cheaper to discover an error using small, well-defined fake data sets than it is to discover the same error using large real data sets.

|  |
| --- |
| **Business-8**: The protocol must support being tested by a suite of automated test cases that can demonstrate that it is behaving correctly. |

A detailed sequence of design decisions that follow from this one are described in the software engineering decisions. However, one of the main side effects of being able to substitute real data for fake data is that the protocol has more potential to be made generically. It forces design to isolate a large part of the program that does not care how specific input data sets were created.

3.4 The Decision to Invest in Making a Reusable Data Set

Originally, the protocol was intended only generate exposure data for a study that was about respiratory health. However, as the use case study progressed, the Imperial team garnered more interest from researchers who wanted to link exposure data for ALSPAC with other kinds of health outcomes. The interest raised the question of whether ALGAE should be designed to generate reusable data sets that would support multiple prospective studies, some of which have not yet materialised.

The idea of making a reusable data set that can service multiple research projects seems to have obvious merit. However, it is important to acknowledge the set of incentives and disincentives that go with making the protocol achieve that goal. In order to examine the value of investing in that effort, we need to first examine the kinds of costs that are associated with doing an activity like the use case study from scratch.

Such a study would likely take years to complete, owing to various sources of delay:

* drafting and refining study design through medical ethics committees
* coordinating a multi-disciplinary team of researchers whose skill sets may not overlap much
* coordinating researchers who would use residential mobility data with cohort administrators who would provide it
* time spent assessing the quality of data and cleaning them
* co-ordinating site visits to work on applying the protocol.

The longer a study lasts, the more prone it is to the natural turnover of staff and the need to spend time recovering institutional memory about low-level details such as the meaning of fields in data sets or of code fragments in programming scripts.

Given the likelihood that the study would require great amounts of work and delay, the benefit of making a reusable data set would be to minimise the effort needed to support a slightly different study. However, we also need to acknowledge the disincentives for making a reusable data set:

* pressure for developers to produce the simplest solution to minimally meet requirements of the study
* a lack of data sharing incentives for the research group who would initially fund getting the results
* reluctance for longitudinal cohorts to allow multiple research groups to link separate studies together

Because developer resources are often scarce in academic project settings, staff programmers are often encouraged to provide the minimal solution in response to short term needs. However, making a reusable data set requires significant forethought and effort to make it appeal to multiple studies.

The data set may have to support fields that might be useful to a future study but which are not critical for satisfying the current one. There is a higher burden to test results because the scope of effect for any error would be multiple studies instead of just one. There is also a higher burden to document the work so that researchers on all the projects can have a shared understanding of what individual result variables mean. All of these factors mean that in the short term, the first study it supports is probably more expensive to make than it needs to be.

The PIs who are funding the work may require incentives to motivate them to share their data sets with other projects. Some of the researchers may be concerned that another project could unfairly benefit from using their results without ample consideration given to matters of credit, publication authorship and cost sharing. Even if it is technically feasible and desirable to produce a reusable data set, researchers may be reluctant to invest in the effort needed to make one if the system of incentives for doing so have not been thought out.

The third deterrent for making a reusable data set is that longitudinal cohorts would likely discourage its use. They often wrestle with two opposing interest: the desire to attract more projects to use the cohort data and the desire to make sure studies do not make their cohort members more identifiable through linked studies.

Suppose Study A links health outcomes related to digestion to life stage exposures of PM10 and Study B links health outcomes related to musculoskeletal systems to the same exposures. Then Study A will likely be able to link with Study B.

Concerns of linking data sets together can be partly allayed by taking various measures. For example, A and B could use different study member identifiers or the exposure values they used could be rounded in different ways. But in practice, it would not take too many exposure variables taken together to increase the ability for the outcomes of A and B to be linked via a common set of exposure values.

Cohort leaders often wrestle with the competing views for encouraging reusable data sets. On one hand, a valuable reusable data set can attract new projects and more funding, all the while making use of results that were paid for by the first study which used them. On the other hand, cohorts are wary of any risk of having collaborators directly or indirectly linking their data to make study members more identifiable.

Issues related to the potential for linking studies together are usually resolved through extensive discussions between the cohort facility and its research collaborators. However, cohort administrators may exhibit an initial wariness about watching one of its collaborators make a data set that could support indirect linking.

Taken together, the three factors will most likely encourage developers to make a minimal solution to support a single study. This way provides the cheapest approach to support the first study that uses the results. It works in the absence of extensive discussion given to how data sharing incentives would work. And it satisfies the interest of showing to data providers that a group is using only the data that are necessary and sufficient for completing their funded activity.

The decision we took to invest in making a reusable data set may have happened because the developer resources came from a shared resource managed by SAHSU rather than from a dedicated resource specifically attached to one study. The database team provides supports for many different kinds of studies. From the perspective of any one research project, the needs of their study would appear unique because they are often focusing on very different domain areas. However, from the perspective of developers, many studies begin to exhibit similar patterns in the features they want in programs.

Despite the three factors mentioned which would normally encourage making a one-off exposure data set, one conclusion was clear: we could not afford to do multiple similar studies from scratch. The interest in anticipating future use was borne from a concern of avoiding repetitious development of protocols.

Once it was established that it was worth the effort to invest in making a reusable data set, the next design question was: “How would we define the attributes of a reusable data set?” Fortunately, the scope the defined where the data set began and ended followed natural divisions in the multi-disciplinary fields which were used in the use case study. We decided to assess exposures for each pollutant for each life stage for each person, and we would generate the results irrespective of the health outcomes with which they could be associated.

|  |
| --- |
| **Business-9**: The protocol would generate a reusable exposure data set that could be linked to the individual and health information stored by the cohort study. |

Now, instead of having a protocol whose end product was a set of associations between exposures and respiratory health variables, its end product was now simply a set of life stage exposure assessments for multiple pollutants. In effect, ALGAE ceased to become a protocol for supporting health studies and became one that supported exposure studies. In the use case study, statisticians would link the exposures to health outcomes as a separate activity to generating aggregated exposure results.

|  |
| --- |
| **Business-10**: Because it produces a data set that will support multiple environmental health studies, the need to support data science and software engineering requirements becomes more important than those concerns would have if the protocol supported only a single study. |

## 3.5 Making a Generic Protocol to Make a Generic Exposure Data Set

In our discussion about the requirements that shaped protocol design, we began with the requirements of supporting a single study that examined the links between early life exposures and later life health outcomes. The protocol was then influenced by business concerns that related to costs, logistics and information governance. In the last section, we arrived at the conclusion that we would make a protocol which used ALSPAC cohort data to produce a reusable exposure data set that would support multiple environmental health studies. It would generate aggregated exposures in a way that was independent of any given set of health outcomes. The protocol would use data from a specific cohort to make a generic data set.

As the use case study progressed, staff scientists began to discover that other environmental health projects which used data from other longitudinal cohorts shared common characteristics. The final major business question that influenced the development of ALGAE was: “Could we use the same protocol but apply it to different cohorts?”

ALGAE’s need to support full automation and testability already seemed to favour developing a generic protocol to make a generic, reusable exposure data set. The elimination of manual work would mean that judgements about data linking and data cleaning would not vary across projects. The need to be able to substitute real data sets for fake data sets introduced a layer of abstraction in design that forced much of the code to not care about cohort-specific data files.

Designing for testability soon caused the code base to be divided into two themes: cohort-specific code that could massage data into standardised input tables and generic code which could be applied to those tables, irrespective of how they were produced. In the main use case study, the developer would create both kinds of code. However, if it were used on another project using another cohort, developers would only have to invest making the cohort-specific code that would produce the correct input tables.

Although the design already seemed to support the idea of making a more generic solution, there were other issues that had to be considered in order to make ALGAE a community resource rather than just a software application that used data from a specific cohort to generate exposure values.

The first main issue is that given the niche area of application for the protocol, we did not expect there would be many users. We also recognised that those projects which could use the protocol would likely exhibit different levels of interest. Rather than invest in trying promote the software as a reusable asset, we decided to tier notions of buy-in. Some groups might want to create their own protocol but would be interested in drawing on our development experiences to inform their own work. Others might be interested in creating input data sets and running the software to generate exposure results. A few may want to adapt the software so that it could support different use cases. And some may be interested in reusing parts of the code to support other projects.

|  |
| --- |
| **Business-11**: Accept that different cohort projects will express different levels of interest in the protocol. Rather than expecting the utility of a generic protocol to be measured by the number of projects that download and use it, anticipated tiered levels of buy-in from other groups. These are: people who want to reuse its design to make their own software; people who want to reuse the software tool with a different cohort; people who want to adapt the code to suit slightly different use cases; and people who want to borrow parts of the code for other projects. |

### 3.5.1 The Need to Provide Documentation to Create Input Data and to Interpret Results

Had the use of the protocol been limited to generating exposure data for the ALSPAC cohort, then Imperial College would likely been able to rely on its own internal documents and in-house project memory to re-run it in other projects. However, if the protocol is meant to run in other projects, then that project memory must be made explicit through documentation that can be understood by people who were not involved with its development. At a minimum, this documentation would have to explain what kind of input data are expected, how to install and run the program and how to interpret the results.

|  |
| --- |
| **Business-12**: The protocol documentation must minimally include three things for prospective users in other projects: an explanation of properties for the input data sets; instructions for installing and running the protocol; and a data dictionary that describes the meaning and context of result variables. |

### 3.5.2 The Need to Accommodate Variations in Input Data between Cohorts

Project developers are responsible for transforming cohort data into the fields expected by the input tables. However, in order to make their jobs easier and thus increase the likelihood that other projects will use the protocol, it makes sense to accommodate some variation in the way input values are represented.

|  |
| --- |
| **Business-13**: Accommodate input values that may have cohort-specific representations but will have the same meaning. |

|  |
| --- |
| **Business-14**: Accommodate different ways of representing input values for yes-no fields. For example, allow ALGAE to understand that “Yes”, “yes”, “true”, and “1” are all ways of representing “Y” for Yes. |

|  |
| --- |
| **Business-15**: Accommodate different ways of representing an empty field value. They may include the empty string “”, a null value, some form of “NULL”, and “NULLIF#”. |

### 3.5.3 The Need to Adapt the Code to Suit Other Projects

In order for ALGAE to become a community resource, it needs to support rather than merely enable new projects to adapt the code to support a different cohort. Ideally, support would come in the form of configurable program options which would allow other researchers to customise the behaviour of the protocol. However, when we considered making configuration options we realised we could create a lot of complexity to make programming options which may or may not be used.

If the developer resources do not exist to provide support for configurable options, the next best approach is to ensure that documentation exists which shows researchers how they can change the code. We identified two major aspects of the protocol that other projects might want to alter:

* changing the life stages
* changing pollutants

Early in the use case study, the epidemiologists had varying opinions about when pregnancy trimesters began and ended. Given that these opinions could vary from project to project or from cohort to cohort, it made sense to carefully identify and comment the code that would be responsible for adding life stages to the database.

|  |
| --- |
| **Business-16**: Document how code could be changed to support different life stages, or different temporal boundaries for existing life stages. |

ALGAE is not entirely agnostic towards the life stages it uses. For the early life analysis there was a need to correct life stage boundaries for study members who had very small values for gestation age at birth. Key to this cleaning process was identifying the life stage which contained the birth date of the study member. It is possible to rename the EL life stage so that is called something else or that it ends at a different temporal boundary. However, there was a need to document this so that cohort researchers could make a more informed decision about changing that life stage:

|  |
| --- |
| **Business-17**: Document any code that may depend on the name of a specific life stage (eg: EL, which normally contains the birth date as a start date). |

ALGAE treats the exposure values for pollutants in exactly the same way. It attaches no special meaning to any specific pollutant; they are all just numeric fields whose values will be aggregated to produce cumulative, average and median values. Another group may want to substitute existing pollutants for new ones or add to the set of pollutants that are used in assessment. Through a thorough process of “search-and-replace”, the protocol code could be altered so that it was supporting a noise emission study instead of an air pollution study.

|  |
| --- |
| **Business-18**: Document how other projects could change the pollutants that are used for the study. |

3.5.4 Generic Considerations for Information Governance Policies

As part of broadening the protocol’s scope of support for other cohorts, we had to examine how the protocol might have to change to accommodate differences in information governance policies amongst them. We had to consider two factors:

* differences in the division of labour that could occur in other partnerships between cohorts and research collaborators
* differences in perceptions of whether a given variable was considered too sensitive to take off-site from the cohort’s facilities

The use case study could have been done in an environment where all or none of the work was done within the cohort’s facilities. The differences in these scenarios could affect a project’s decision to encrypt or anonymise certain fields. If input data are produced and linked entirely within an isolated computer network, there may be benefits in not anonymising certain fields. For example, if a project could use geocodes that contained map coordinates, it could use that information to help identify some reasons why some of them were not generating exposure data (eg: out of bounds). As well, researchers could use geocodes for two successive address periods to determine whether they represented distinct locations or if the second address was a correction of the first.

Similarly, if all the work happened within an isolated network managed by the cohort, then there may also be value in using cohort member IDs rather than using study member IDs that were generated specifically for one study. Using cohort member IDs could make it easier for cohort staff to link the records with other data sets they maintain about their members.

In both cases, the impact of accidentally disclosing more identifiable data is higher but the likelihood of it happening could be very low because the data are all physically contained within the cohort’s secure facilities. Two main benefits of using more identifiable data are that they can make data linking less error prone and it can help make it easier for researchers to explain anomalies in the results.

Because some projects may want to anonymise fields in different ways and others may not want to encrypt some fields at all, we decided it was best to not make encryption or anonymization tasks that the protocol would do. Instead, we expect that if projects want to encrypt person\_id and geocode fields, they should do it as part of producing the input data sets. ALGAE attaches no meaning to values for either of these fields, and just uses them to link records from different tables together.

|  |
| --- |
| **Business-19**: The protocol will not support features for anonymising or pseudonymising fields. Projects that want to de-identify these variables should do so as part of the process they follow for creating the input data files or they should de-identify variables after the results have been generated. |

|  |
| --- |
| **Business-20**: The protocol will not attach any meaning to person\_id or geocode fields, and will just treat them as identifiers that link tables together. |

There is one way that the protocol can assist projects where most of the sensitive data is being linked at the sites of partnering research institutions instead of at the cohort’s facilities. Modern information governance policies adopted by research institutions usually compel a data destruction provision, which compels researchers to destroy sensitive data when they no longer need it.

|  |
| --- |
| **Business-21**: In order to support data destruction provisions in information governance policies, the protocol will write all of its result tables to CSV files so that projects can safely delete the entire database which produced them. Isolating results in this way allows projects to minimise the risk of holding sensitive data in the intermediate data sets produced by the protocol. |

The next information governance consideration is determining how protocol design should handle differing opinions about whether a given result variable is too identifiable to be taken off-site from the cohort’s facilities. In the use case study, none of the results included geocodes or birth dates. None of the variables that are included in the results should pose a significant risk of identifying study members. However, in other projects, perceptions of risk for identifiability may be influenced by preferences of cohort administrators and by the amount of other data about cohort members that a research group may intend to link with exposure results.

There is no way for us to anticipate how any cohort administrators from any given cohort will decide which variables are deemed safe to take off-site. Therefore, the protocol design will ignore information governance concerns about identifiability in variables. The risk of having enough variables to identify cohort members will instead be mitigated in three ways that are managed by the research projects:

* pre- and post-processing activities that de-identify data;
* a vetting process cohort administrators can use to determine if data for a given result field can be taken off-site;
* changing aspects of physical security that would manage sensitive data linking activities

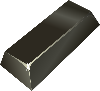
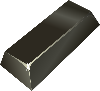
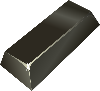
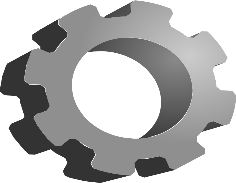
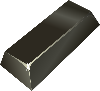
|  |
| --- |
| **Business-22**: The decision about which fields the protocol will produce are based on what fields would be useful for the scientific use case and are not based on the information governance policies of any one cohort. It is possible that in a given project, ALGAE produces more data than what is necessary and sufficient for the research purpose. It is also possible that it will generate more data than a given cohort will allow to be taken off-site. |

### 3.5.5 The Need to Support an Open Source Community

If ALGAE is to become a community resource that can be used in other projects, researchers should be able to obtain, modify and run the protocol without having to rely on the protocol’s original authors. Open sourcing makes the protocol more transparent and easily maintainable my multiple programmers who may be attached to different projects:

|  |
| --- |
| **Business-23**: The ALGAE protocol should be open sourced through a standard OSI license. |

**4. Data Science Decisions**



Turning results into knowledge, auditing, data cleaning,

data linking, sensitivity analyses, repurposing data sets

## 4.1 The Meaning of Data Science for ALGAE

Data science is a term that carries multiple meanings. Some definitions emphasise the synthesis of data analysis, computer science and statistics. Others emphasise using statistical analyses to identify trends in large data sets that contain semi-structured or unstructured data. Common to most definitions is the notion of a set of techniques that can be used with data, independently of the domains to which they are applied.

In the ALGAE Protocol, data science is about making analyses that use the scientific method to derive valuable knowledge from data sets which may have to be repurposed. Before we discuss the sequence of data science decisions which produced the protocol, it is worth elaborating on three terms that are contained in our definition: scientific method, valuable knowledge and repurposed.

For our discussion about data science design decisions, we will introduce them in roughly the order we would encounter them in a simple workflow: preparing input data, processing input data and generating results. Much of the first part is about how cohort staff would need to prepare the data so that ALGAE can add value to the results. The main issues relating to input data are including data quality flags that may depend on cohort-specific data quality issues and in making the conventions for naming and locating input files clear.

Design decisions for processing input data cover issues such as: preserving the provenance of original data; policies that determine how deleted data are expressed; supporting traceability of intermediate results back to original data sets; supporting sensitivity studies by identifying subsets of results that were affected by changes in the data; and how the influence of those changes on results could be expressed through a form of error measurement.

In the final part of the discussion, we cover how we can add value to results. The main areas include: supporting multiple methods of assessing exposure to help gauge the effects of error measurements on results and to make results compatible with older studies; the handling of null values; and supporting naming conventions for naming output files and output variables.

## 4.2 Adding value to input data

### 4.2.1 Supporting version and is\_valid fields in the geocode input data

The most important core aspect of the ALGAE protocol is how it uses residential address histories to support high resolution exposure assessments, even though the mobility data were originally gathered to audit current addresses rather than track past ones. The difference between these two goals may account for why the input data may result in a significant number of residential addresses that cannot be geocoded, and address periods that exhibit missing values, gaps or overlaps in the residential address histories.

The first step cohort projects may take to use ALGAE is to extract a list of residential address histories for study members. They would then embark on an iterative process for associating each residential address with a set of map coordinates that could provide inputs to software that produced historical exposure values. For the first iterations, projects would likely convert addresses expressed by a number of database fields (eg: address line 1, address line 2, city, post code) into a single canonical field value that was amenable to being processed by geocoding software.

Geocoding will undoubtedly fail to match some canonically represented addresses with map coordinates. There are many reasons why these failures occur. Examples include: typographical errors; partially specified addresses; addresses that use deprecated post codes; addresses that use post codes that are unknown to the geocoding software.

Whereas a tracking system might anticipate the need to have addresses that were machine readable by GIS software, a system that only needs to audit addresses for cohort mail-outs may only need addresses that are readable by mail handlers in the post office system. The difference in data quality of addresses produced through these different contexts could significantly affect exposure assessments.

Even when geocoding software is able to produce map coordinates, they may represent other problems:

* the locations may not make sense (eg: centroids of postal codes that are over inhospitable areas such as bodies of water or mountain tops)
* the locations are based on poor matches (eg: the middle of a very long road)
* the locations could be outside the exposure area

The need for human judgement to be applied where machine judgement fails implies that the geocoding activity may involve multiple iterations. Making corrections may involve manually matching addresses with locations, or it may mean using different geocoding software applications. The techniques for geocoding addresses could vary from one project to another.

ALGAE does not need to care why a geocode may be invalid; it just needs to know whether the value is good enough to place someone at a location for a given period. It needs a generic data quality flag to help it assess exposures:

|  |
| --- |
| **Data Science-1**: The input CSV file that holds data about geocodes will have a yes/no flag called “is\_valid”. If the flag value is Y, then the geocode is considered valid and may be used in exposure calculations. If the flag value is N, then the geocode is considered not valid, and may cause some study members to be excluded from exposure assessments. The criteria for validity are defined by individual cohorts. |

The quality of geocoded results may vary from one iteration to another. The difference could owe to preferring one geocoding application over another, or favouring the work of one group of researchers over another in cases of manual geocoding. Researchers may want the ability to link the geocodes found in residential addresses with the iteration that made them. In order to facilitate this, we decided to include a concept of a version in the geocode input data:

|  |
| --- |
| **Data Science-2**: The input CSV file that holds data about geocodes will have a free-text field called “version”. Cohorts can use the field to describe an iteration of geocoding that uses a particular version of software, was done by a particular method or was done by particular people. |

### 4.2.2 Supporting data quality flags relating to coverage of exposure periods by address histories

Remembering that the residential address histories come did not come from a tracking system, we need to develop flags that help us judge how well the addresses cover the exposure period of interest. In the use case study, cohort members were recruited when their mothers were already pregnant with them. The first address in the Contacts Database would have been the address their mothers occupied when they enrolled them in the birth cohort. Because the early life analysis begins at the date of conception, it is possible that the mothers may have lived somewhere else sometime between that date and the date of enrolment.

In other birth cohorts, it is possible that women would be recruited before they became pregnant. In this type of study, we assume that the first recorded address could predate the conception of cohort members they may produce.

In order to guarantee that the protocol can assess exposures for the early weeks of pregnancy, the protocol will ensure that the start date of the first address period for each study members is at least the value of the conception date. It makes an implicit assumption that the mother didn’t move between the conception date and enrolment date of her child.

However, researchers may want to know how many study members would have actually been at their first recorded address during conception. That information would likely come from a collection of cohort-specific variables. For example, the questions could be of the form: “Have you lived at your current address during your entire pregnancy?”, “How many times have you moved in the last X years?”, “Have you lived at other addresses in the past year?”

In order to support a generic design across cohorts, ALGAE would need to not care about how or why this assumption could be shown to be false. Instead, it would try to preserve it as a sensitivity variable on behalf of researchers who were examining results.

|  |
| --- |
| **Data Science-3**: The input CSV file that holds data study member data will have a yes/no field called “at\_1st\_address\_conception”. If the flag value is “Y”, then the mother of the cohort member lived at this address at the date of conception. |

The Contacts Database was designed to maintain the current postal addresses of cohort members so that the cohort administrators could contact them. It is possible that some addresses could have just been used for forwarding addresses, while members were living somewhere else. Homelessness, time spent in hospitals or prisons, or time spent abroad could be among the reasons why some study members would have lived at addresses that were not recorded in their residential address histories.

The cohort researchers would have to decide what criteria would be used to judge whether a study member had spent significant time during their exposure time frame at a location that did not appear in residential history records. In order to allow it calculate exposures, the protocol will assume that study members lived only at addresses that are specified in their residential address histories. However, again the protocol will require a data quality flag to help researchers determine how well that history covers the exposure periods of interest:

|  |
| --- |
| **Data Science-4**: The input CSV file that holds data cohort member data will have a yes/no field called “absent\_during\_exp\_period”. If the flag value is “Y”, then researchers can be confident that a cohort member spent a significant amount of their exposure time frame living at a location that was not specified in his/her residential address history. |

### 4.2.3 Supporting naming conventions of input data files

In order to make it easier for researchers to locate input files, ALGAE will require cohorts to use naming conventions for both the input files and the variables within those files. We assume that early life and later life analyses will each warrant their own separate input files, even though they are likely to have great overlap. The following directory structure will be used to store the inputs and outputs of the protocol for each analysis. Note that all input tables will be marked with “original\_”.

early\_life

input\_data

**original\_study\_member\_data.csv**

**original\_geocode\_data.csv**

**original\_exposure\_data.csv**

**original\_address\_history\_data.csv**

results

later\_life

input\_data

**original\_study\_member\_data.csv**

**original\_geocode\_data.csv**

**original\_exposure\_data.csv**

**original\_address\_history\_data.csv**

results

The original\_study\_member\_data table will contain the following fields: person\_id, comments, birth\_date, best\_gest, absent\_during\_exp\_period, at\_1st\_address\_conception. The original\_geocode\_data table will contain the fields: geocode, version, comments, ed91, oa2001, coa2011, has\_valid\_geocode. The original\_exp\_data table will contain the fields: geocode, comments, date\_of\_year, pm10\_ rd, nox\_rd, pm10\_gr, pm10\_tot. The original\_address\_history\_data table will contain the following fields: geocode, comments, start\_date, end\_date. The role of the comments flag in each input file will be discussed in the Software Engineering decisions.

|  |
| --- |
| **Data Science-5**: Cohort studies will need to comply with naming conventions for both the paths of input files and for the variable names within those files. |

## 4.3 Adding value to intermediate results

### 4.3.1 Preserving the provenance of input data and intermediate results

Now that we have defined the names of tables and table fields that ALGAE expects for input, we can discuss how we can preserve both the input data sets and the intermediate result sets that are derived from them. Part of the value of the protocol will lie in its ability to let researchers trace results back to the original data sets which produced them. Some of the design decisions we take to satisfy this data science requirement are also discussed in the software engineering decisions section.

If the results appear to be reasonable, then many scientists may not care about the various intermediate states that data had as it was being transformed from input files into results. However, it is when the results look like they are in error that the researchers will want to have a means of tracing the steps made by the protocol leading from original data to results.

The first step we can take to support traceability is to preserve data from the CSV files in original data tables, and then let all subsequent data transformation be applied to staging tables, which are copied versions where field values that may vary from one cohort to another are all standardised. By copying the original data rows rather than trying to update them with changes, we can help preserve the original states of data that may explain any data anomalies in the results.

|  |
| --- |
| **Data Science-6**: Preserve the provenance of data in the original data files by storing unaltered data in original data tables that are not touched by the rest of the protocol. Let the protocol apply its data transformations on staging tables, which will contain versions of the original data that have standardised the representation of field values. |

The second step we can take in supporting traceability is to favour using sequences of temporary tables to represent complex data transformations rather than having a few complex monolithic queries to do the same activity. As we will see in the Software Engineering Decisions, the latter approach is probably more efficient than the former approach in terms of how the software will process large data sets. However, temporary tables that capture progressive states of data transformation make it easier for researchers to trace the source of errors.

|  |
| --- |
| **Data Science-7**: In order to make it easier for researchers to identify errors, favour expressing complex operations through a sequence of simple temporary tables rather than by using a single, complex monolithic query to do the same work. |

In ALGAE, sensitivity variables and cleaned address periods are built up through a succession of temporary tables. Each successive table preserves changes made in the previous one. In some operations, multiple actions may cause original field values to change. For example, the start date for an address period appears as the fields start\_date, adjusted\_start\_date and fin\_adjusted\_start\_date. Each field represents a stage in the protocol where a value may change.

|  |
| --- |
| **Data Science-8**: Save changes using modified copies of table fields rather than changing original ones. |

Next, we can help support linking intermediate results with original data by preserving the location that records would have had in the original input files. For example, data from the original\_address\_history\_data.csv file are changed through a succession of over a dozen temporary tables. These tables preserve the original row number of each record so that researchers can find the row in the CSV file that provided a specific address period.

The original row numbers of input data will be dropped once intermediate results depend on combining data from multiple tables. However, at least when data anomalies related to address periods are examined, there will be a clear link between a transformed record and an original record.

|  |
| --- |
| **Data Science-9**: Capture the original row number of records from input files in corresponding staging tables. Where it is possible to do so, promote the original row number fields through temporary tables that may use them. |

### 4.3.2 Filtering data using flags rather than using deletion or omission

Next, we need to decide on a policy for dealing with filtering data. Often it makes for simpler code and for better performance if poor quality records are either deleted from a table or simply left out when records are copied from one temporary table to the next. However, this approach can make it more difficult to audit changes, or to assess the extent to which they affected results.

For example, the protocol may identify duplicate address periods, where the period of one is completely subsumed by the period of the next. It would be easy to simply leave out the duplicate records from subsequent temporary tables. But in the final version of the address periods, the researcher may lose an appreciation for what records were deleted for each person. In order to preserve the provenance of filtering operations, it is better for the scientists if bad records are flagged as deleted rather than actually be deleted. Where possible, both records that have been used and records that have been ignored should appear in final results.

|  |
| --- |
| **Data Science-10**: Wherever possible, do not delete table rows. Instead, flag them as being deleted. This approach allows protocol users to inspect the nature of deleted data identified for each study member. |

|  |
| --- |
| **Data Science-11**: Promote ignored rows through successive temporary tables but use deletion flags to ignore them from consideration in calculations. |

### 4.3.3 Repurposing residential address histories

The way that the residential address periods are processed must recognise that they were gathered for one purpose and used again in another. In the use case study, the residential address histories were derived from a Contacts database that cohort administrators used to maintain the current contact details of its cohort members. The original data set that we used comprised the following fields: a study member id, a geocoded address, a start date and an end date.

Transforming the administrative data into research data involved mapping concepts as well as cleaning fields. The start date was derived from the time stamp in the database that indicated when a study member’s contact details were updated. The protocol design makes an important assumption here:

|  |
| --- |
| **Data Science-12**: Assume that the date that a study member’s current address was updated in the contacts database is the date when he or she began living at a new address. This value is the start date of an address period. |

It was not always clear how the end dates were calculated. In many cases, the end date of an address period equalled the start date of the next address period. This pattern suggested that the Contact database application automatically calculated the end date based on the date when the next entry was created. However, in other cases, successive address periods exhibited gaps and overlaps, suggesting that at one time, some of the end dates may have been manually edited. It is not clear why these anomalies occurred, but we speculate that the application may have begun as a simple table editing tool that later evolved into a more complex application, and then was migrated at least once to a new database management system.

Almost every address period had a start date, but many of them were missing an end date. Again, we can only speculate on the reason why some of the end dates were missing. It may have been the case that some were left blank because the study member had not since moved from the location in that address period.

So that it could produce a temporally contiguous address history, the protocol had to ensure that blank start date and end dates would be imputed. We made the following assumptions about how blank values should be filled in:

|  |
| --- |
| **Data Science-13**: Ensure that each address period has non-blank values for its start date and end date fields. Impute missing start dates with the conception date of the study member. Impute missing end dates with the current date. |

Once all of the address periods had non-blank values for start and end date fields, they needed to be chronologically ordered in a consistent way:

|  |
| --- |
| **Data Science-14**: Order address periods in ascending order, first by person\_id, second by start\_date and third by duration. |

Ordering address periods revealed gaps and overlaps between them. In order to fix these problems, the protocol needed to adopt a consistent policy for adjusting the end date of one address period and the start date of the next. The policy was informed by assumptions we made about which of start or end dates provided the more reliable signal:

|  |
| --- |
| **Data Science-15**: Assume that in the residential address histories, start dates are stronger signals than end dates. Assume that study members were likely already living at the geocoded location in an address period at or before its start date. |

|  |
| --- |
| **Data Science-16**: In the process of fixing temporal gaps and overlaps between successive address periods, act to preserve the start date of the current address period over the end date of the previous one. |

The algorithm for fixing gaps and overlaps became:

|  |
| --- |
| **Data Science-17**: Let an and an+1 be two successive address periods. If a gap exists between them, then let an+1.start\_date = an.end\_date + 1 day. If an overlap exists between them, let an.end\_date = an+1.start\_date – 1 day. |

In the algorithm, a duplicate address period was a special case of temporal overlap:

|  |
| --- |
| **Data Science-18**: Let an and an+1 be two successive address periods. an is considered a duplicate if it is temporally subsumed by an+1. If this is true, then an will be flagged as a deleted record. |

Once the address periods were adjusted so they could form a temporally contiguous record showing movement patterns, the address history records had to be made to fill up any remaining unaccounted days in the exposure time frame:

|  |
| --- |
| **Data Science-19**: Ensure the address history for any study member spans the entire exposure time frame (eg: [conception\_date, birth date + 1 yr – 1 day]). Ensure that the start date of the earliest address period is adjusted to be at least as old as the conception date. Ensure that the end date of the latest address period is adjusted to be at least as early as the last day of the exposure time frame. |

### 4.3.4 Fixing bad geocodes

In the use case study, we had to do both temporal and spatial data cleaning for address periods. The protocol requires that a valid geocode that has at least some exposure records exists for each address period. However, we identified a significant number of bad geocodes. A geocode is bad if:

* it is an empty field value, meaning that geocoding failed to resolve coordinates
* it is not an empty field value but the is\_valid is “N”, usually meaning that although coordinates were produced, the match should be considered too poor to use
* it is a valid geocode but it has no exposure values associated with it

In order to simplify calculations, we had to make a decision about how to handle study members who had an unknown exposure for some part of their exposure time frame:

|  |
| --- |
| **Data Science-20**: If a study member has at least one address period which has a bad geocode and which spans part of their exposure time frame, he or she will be excluded from further exposure assessments. |

Unfortunately, many study members met these criteria. In order to help increase the number of study members who could have their exposures assessed, we tried to look for patterns in the residential address histories which we might be able to fix.

The Contacts database application that provided the histories treated each new update as a separate time-stamped record, whether it represented a new address or a correction to the current one. We assumed that the following criteria would identify an address period whose geocode could be fixed:

|  |
| --- |
| **Data Science-21**: If address period is fixable if it meets the following three criteria:   1. It has a bad geocode 2. It is immediately followed by an address period which has a valid geocode 3. It has less than 25% overlap with any life stage |

The scenario associated with these criteria is one where a study administrator observed an error in the address details and fixed it. We would then assume that the initial incorrect address and the subsequent corrected address would refer to the same location. The correction algorithm would become:

|  |
| --- |
| **Data Science-22**: Let an address an be a fixable address period. The protocol will attempt to ignore an from further calculations by subsuming it within an+1: an+1.start\_date = an.start\_date. |

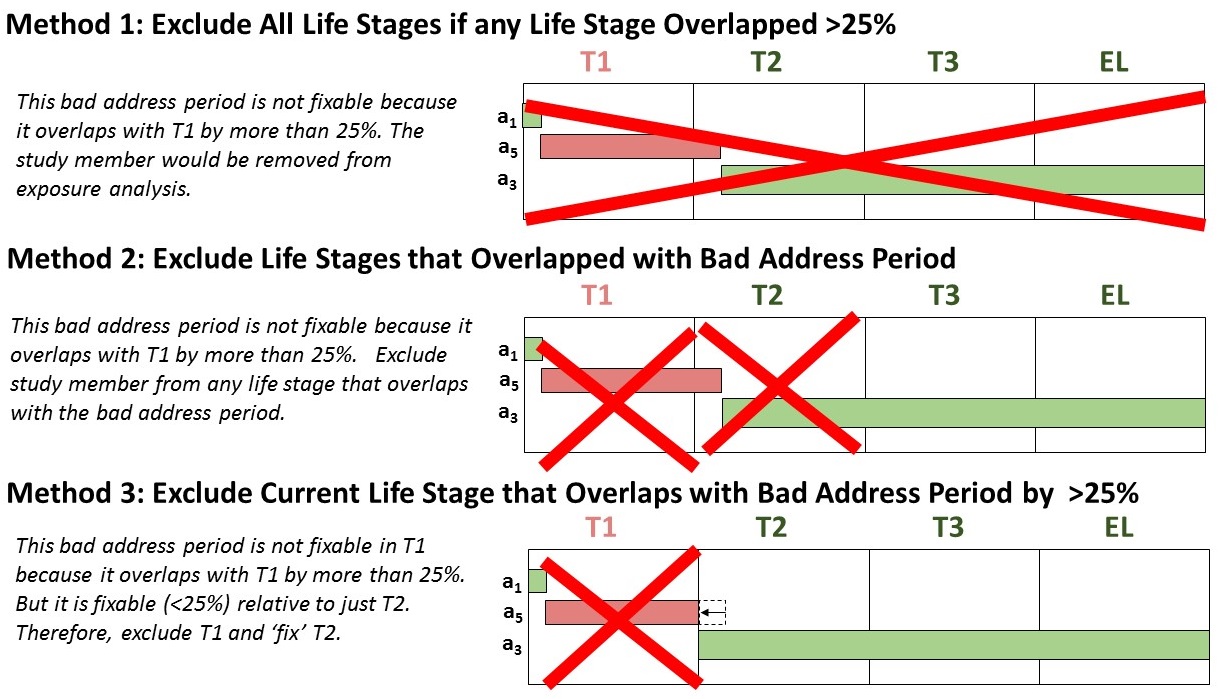
The protocol would mark address periods that were fixable with a flag and ignore them as if they never existed in the residential address history. Note that a fixable address period that is ignored is different than an address period that is valid but which is deleted as a result of adjustments made to fix gaps and overlaps.

### 4.3.5 Policy for Handling Study Members with Unfixable Bad Address Periods

The next major design decision that follows from our discussion about fixing bad address periods is: How ALGAE should handle study members who have relevant bad address periods that can’t be fixed? For example, suppose study members occupy an invalid geocode for 80 days out of 92 days in T1. The bad address period will not be fixed because it occupies more than 25% of some life stage. How should the protocol handle that?

Initially, the protocol was made to exclude study members who had ‘unfixable’ bad address periods in their exposure time frame. This approach was adopted to simplify coding and testing. By filtering out study members who had null value exposures for certain parts of their exposure time frames, the rest of the protocol code could assume that exposures would be non-null. Null values are often problematic in programming because they can cause errors in code that does not anticipate them.

However, this approach ended up excluding too many people. Next, we considered excluding only those life stages that overlapped with an unfixable bad address period. However this too was problematic in cases where the overlap would be only a couple days in a much longer life stage. The extent of overlap was still important to consider, and we wondered if we could somehow apply the fixing algorithm on a life stage basis. In this approach, we would try to apply the fixing algorithm so that a bad address period was fixable if it was less than 25% of the current life stage rather than any life stage. The following diagram illustrates the progression from considering the first approach to considering the third:



Implementing Method 3 proved problematic because the decision to acknowledge or ignore the contribution of a bad address period relative to each life stage could have knock-on effects on the way address periods would be subsequently cleaned. Cleaning could cause a slightly different address history for one life stage than it would for another. In theory, we might need to maintain n address history tables, one for each of n different life stages. Because exposure assessment is critically dependent on an address history table, multiple variants of address histories would inevitably lead to multiple variants of exposure assessments (ie: cleaned mobility, uncleaned mobility, life stage and for early life the birth address approach).

None of these ideas proved fruitful, yet we needed a way to be more forgiving for address histories that may feature bad address periods. An alternative solution became practical to implement when an unrelated decision was made for the code. It became clear that in order to simplify the code base and testing, that early and later life analysis would both be made to work based on daily exposure values. Once early and later life analyses shared the same code to aggregate daily exposure values, a lot of code could be removed from the code base (See Section 4.3).

Using daily exposure values meant that the only functions that would have to be able to cope well with NULL values were the SUM(), AVG() and median() functions. SUM() and AVG() are built-in postgreSQL functions, and the implementation of the median() function appears to be widely used and well tested. All three of these methods were found to ignore NULL values in a predictable way. Allowing null daily exposure values to propagate through analysis was no longer problematic, because the aggregation functions themselves were designed to cope with that scenario. If study members occupied the first ten days of T1 at an invalid geocode, it would just mean that SUM(), AVG() and median() would have a list of values where the first ten values were null.

However, even if the aggregation code could cope well with null values, the scientists would require some breakdown of how many days in a life stage were spent at bad address periods. We assume that bad exposure values could vary from one pollutant to another. For example, we assume that daily exposure data could include an extensive set of pm10\_tot values but lack nox\_rd values for certain days.

|  |
| --- |
| **Data Science-23**: Assess the data quality of exposure values for each pollutant independently of one another. |

For each pollutant, we developed five data quality variables that could help researchers assess the quality of life stage exposures. These mutually exclusive categories are:

* **invalid address day**: the geocode for the address period is invalid and there are no exposures associated with it.
* **out of bounds address day**: the geocode for the address period is valid but there are no exposure values associated with it for that pollutant.
* **poor match day**: the geocode for the address period represents a guess for a very poorly specified residential address period. Even though it may be used to generate exposure values, it is marked as having an invalid geocode.
* **missing exposure day**: the geocode for the address period is valid and there are daily exposure values. However, for specific days, there are no exposure values available.
* **good match day**: the geocode for the address period is valid, and there is an exposure value available for a given day.

These variables are expressed as day counts that together should add up to the number of days in the life stage. With this new feature, the life stage exposure of pm10\_tot in T1 could be associated with a count of 90 days of good match days and 2 invalid address days. Scientists can use these day count variables and the life stage duration to make their own decisions about what results are good enough to use in analysis.

|  |
| --- |
| **Data Science-24**: For each pollutant, exposure results will be accompanied by day count variables that describe various mutually exclusive kinds of daily exposure values that were used in aggregation. These mutually exclusive categories include: invalid address days, out of bounds days, poor match days, missing exposure days and good match days. The sum of these counts will be the same value as the life stage duration. |

Note that for the same pollutant and life stage, these counts may vary from one exposure assessment method to another. ALGAE enhances daily exposure records so that each pollutant comes with its own set of these counts. Each assessment method may choose different daily exposure records, but however they choose the records, the count variables are aggregated alongside the exposure values themselves.

### 4.3.6 Supporting sensitivity variables

When researchers analyse the results, they may want to understand the influence of data cleaning actions and data quality attributes on trends they may observe. For example, if they are particularly interested in examining Trimester 1 exposures, they may want to compare the associations they make using all study members with associations they make for study members who were definitely living at their enrolment address when they were being conceived. They may want to ignore study members whose address periods needed significant cleaning, or they may want to know whether members who moved during pregnancy showed higher or lower exposures than those study members who did not.

The protocol needs to support a set of variables which quantify the ways input data are transformed to make results and which help researchers assess the confidence they can have in making assumptions that govern important calculations:

|  |
| --- |
| **Data Science-25**: The protocol will support a set of sensitivity variables that will allow researchers to quantify the influence of data quality attributes and data cleaning actions on subsets of exposure results. These variables will provide meta data about data transformation activities and will be provided as a result file for users. |

We now move on to identify themes of variables that would be useful in supporting sensitivity studies. The first two come from variables that already appear in the original\_study\_member\_data table. We assume that in most projects, researchers will have access to results but not to the input data. Therefore, if the input variables would be valuable for sensitivity analyses, then they need to be copied to the set of sensitivity variable results:

|  |
| --- |
| **Data Science-26**: The sensitivity variables will include variables that allow researchers to isolate exposure results based on how confident they can be that the residential address histories can cover the exposure time frame of interest. These variables will borrow the at\_1st\_addr\_conception and absent\_during\_exp\_period variables from the original\_study\_member\_data table. |

In the use case study, the exposure scientists were trying to understand why some of the exposures for Trimester 3 were very small. It later became clear that many of these low life stage values were due to the short Trimester 3 period that occurs with premature births.

|  |
| --- |
| **Data Science-27**: The sensitivity variables will include ones that can help researchers determine whether low exposure values for Trimester 3 are due to a premature births or to areas that exhibit low levels of pollution. It will borrow the estimated\_gestation\_age field from the original\_study\_member\_data table, and include a yes/no flag is\_gestation\_age\_imputed. |

The effect of poor quality geocodes can significantly affect exposure results. If a study member has at least one address period that covers part of the exposure time frame and has a bad geocode, then he or she will be excluded from exposure assessment. It is important that researchers have the ability to understand more about why some study members would not have exposure assessments:

|  |
| --- |
| **Data Science-28**: The sensitivity variables will include ones that can help researchers identify study members who were affected by bad geocodes. These variables will include the following totals measured throughout the whole exposure time frame: invalid\_geocodes, fixed\_geocodes, out\_of\_bounds\_geocodes, and has\_bad\_geocodes\_within\_time\_frame. |

For study members do have exposure results, the researchers will want to know which of them were greatly affected by data cleaning changes made to their address period data. The most important types of change in address periods are imputations that are made to start and end dates.

|  |
| --- |
| **Data Science-29**: The sensitivity variables will include ones that can help researchers assess the extent to which the field values of address periods were imputed for each study member. These variables will include the following totals measured across the whole exposure time frame: imp\_blank\_start\_dates, imp\_blank\_end\_dates, imp\_blank\_both\_dates. |

The next most important group of changes are adjustments that are made to the start and end dates of ordered address periods to make them temporally contiguous. The changes would describe the number of address periods that spanned part of the exposure time frame and exhibited: gaps, overlaps and deletions. As well, they may want to know the total number of address periods that study members occupied during their exposure time frame. The total could be used to identify whether exposure trends were related to mobility patterns.

|  |
| --- |
| **Data Science-30**: The sensitivity variables will include ones that can help researchers assess the kinds of data cleaning changes that were made in order for the protocol to make a temporally contiguous record of movements that cover the exposure period. These variables will include the following totals, measured across the whole exposure time frame: total\_addr\_periods, gaps, gap\_and\_overlap\_same\_period, over\_laps, deletions. |

The researchers may also want to have a measure of the total number of days that were changed across all address periods. For example, consider two successive address periods:

a1: [MAY-01-1995, MAY-05-1995] 🡺 [MAY-01-1995, MAY-03-1995] (2 days changed)

a2: [MAY-04-1995, MAY-07-1995] **🡪** [MAY-04-1995, MAY-07-1995] (0 days changed)

Total for a1 and a2: 2 days changed.

Note that in cases where there are multiple successive overlaps, some days could be counted more than once:

a1: [MAY-01-1995, MAY-05-1995] 🡺 [MAY-01-1995, MAY-03-1995] (2 days changed)

a2: [MAY-04-1995, MAY-10-1995] **🡪** [MAY-04-1995, MAY-04-1995] (6 days changed)

a2: [MAY-05-1995, MAY-20-1995] **🡪** [MAY-05-1995, MAY-20-1995] (0 days changed)

Total for a1, a2, a3: 8 days changed. Note that MAY-05-1995 is counted twice in the total.

|  |
| --- |
| **Data Science-31**: The sensitivity variables will include days\_changed, which is the total number of days that were adjusted across all address periods which overlapped with the exposure time frame. The total considers the total number of days that each address period was shifted in data cleaning. In cases of successive overlaps, some days may be counted more than once. |

When we cover the topic of measuring exposure measurement error, we will cover the concept of total\_contention\_days. A contention day is one where gap and overlap errors in residential address histories would allow a person to be placed at more than one location on a given day.

|  |
| --- |
| **Data Science-32**: The sensitivity variables will include total\_contention\_days, which measures the total number of unique days in an exposure time frame that were involved with a gap or an overlap. It is called a contention day because on such a day, a study member could be placed at either of two locations: the location that data cleaning assigned in a cleaned set of address periods, or the location that may have been allowed in uncleaned address periods. |

Note that in the example of three address periods we just covered, days\_changed would have a value of 8, but total\_contention\_days would have only 7, because it would only consider the total number of unique calendar dates that had to be adjusted.

Another factor which could influence exposure results are the number of days in an exposure time frame that are associated with exposure values, but which may be missing values for some days. The original\_exposure\_data table should contain enough records to span the entire exposure time frame of interest. However, if the exposure modelling software missed producing data for some days, then it could cause some study members to have underreported exposures.

|  |
| --- |
| **Data Science-33**: The sensitivity variables will include missing\_exposure\_days, which gives the total number of days in the exposure time frame that are not associated with exposure values. |

These sensitivity variables will let researchers isolate study members who have been affected by poor data quality values or by data cleaning changes. However, it is likely that researchers will lose too many people if they exclude all study members whose records had to be imputed or adjusted. It is not sufficient for them to know that they were affected by changes. The researchers will want to know how much the changes would affect the results. Gaps and overlaps in the residential address histories might cause significant exposure misclassification error by placing study members at one location when they may have been at another. The protocol needs to have a means of associating error bars with exposure results.

### 4.3.7 Developing an approach for assessing exposure misclassification error

In the use case study, exposure scientists assumed that the exposure area could show high levels of spatial and temporal variation. The epidemiologists assumed that exposures assessed over very short life stages could have a significant impact on health outcomes later on in life. It also then follows that if potentially small exposure values measured in these life stages could make a difference in health outcome analysis, so too would slight variations between them. ALGAE makes the following assumption that initiates our discussion about assessing exposure misclassification error:

|  |
| --- |
| **Data Science-34**: Assume that on a given day, placing a study member at one location when they actually lived at another could present an exposure misclassification error that was significant for studies that used the life stage exposure results. |

The exposure measurement error that can occur due to gaps and overlaps can be expressed as a kind of opportunity cost, where a person was placed at one location instead of another. The calculation for exposure measurement error is calculated as:

|  |
| --- |
| **Data Science-35**: Suppose the protocol cleans two successive address periods an and an+1, where a gap or overlap exists between them. Let the exposure at the location assigned by data cleaning to correct a given contention day be called the assigned exposure. Let the exposure at the other location the study member could have occupied on that day be called the opportunity cost exposure. The exposure classification error for that day of contention will be calculated as |assigned exposure – opportunity cost exposure|. |

Now that we have a way of calculating the exposure misclassification error for a single day, we can aggregate these values in the same ways that are used for exposure values:

|  |
| --- |
| **Data Science-36**: Daily values for exposure measurement error will be aggregated in the same way as exposure values are aggregated. For example, error values will be expressed for each pollutant for each life stage for each study member. Error values will be further reported through and use average, sum and median aggregation operations. |

## 4.4 Adding value to results

Once the protocol is able to place study members at exactly one location for each day of their exposure time frame, it is then able to assess both the daily exposure and exposure misclassification values for each day as well. The availability of such high resolution exposure data allows the protocol to assess exposures in a way that considers all the places they may have occupied during that time frame of interest. However, there are at least two reasons why scientists would want the protocol to calculate less precise versions of exposures as well.

First, they may want to know how much difference there is in the results between a high resolution exposure assessment and a low exposure assessment that use the same input data sets. Early in the use case study, scientists wanted to know whether the gaps and overlaps that would be fixed by the protocol made a significant difference to results compared with alternative assessment methods that used more generalised exposure proxies.

Second, they may want a means of analysing the data using an assessment method that makes it more comparable with other studies. Generating exposure values in the same way as many other projects allows scientists to better set their work in the context of existing bodies of work.

### 4.4.1 Supporting multiple ways of assessing exposures for the same life stages and pollutants

Thus far, we have only discussed having one method for assessing life stage exposures. It cleans the residential address histories and then, for each study member, it considers the contributions of exposures from all address periods that overlap with his or her exposure time frame. Average, cumulative and median exposures are calculated for each pollutant of each life stage of each study member. As well, exposure misclassification errors are aggregated in the same way as exposure values. We will call this the cleaned mobility method:

|  |
| --- |
| **Data Science-37**: The cleaned mobility assessment considers exposure contributions from all cleaned address periods that overlap with the exposure time frame. Daily exposure measurement errors are calculated to correspond with daily exposures and they are aggregated in the same ways. |

In the use case study, the exposure scientists wanted more ways to determine the effect of data cleaning errors on results. They suggested two ways, which are described as follows:

|  |
| --- |
| **Data Science-38**: The uncleaned mobility assessment considers exposure contributions from all cleaned address periods that overlap with the exposure time frame. However, it omits any days that spanned a gap or overlap error in a study member’s residential address history. No exposure measurement errors are assessed in this approach. |

|  |
| --- |
| **Data Science-39**: The life stage mobility assessment uses the locations study members used on the first day of one of their life stages to represent the location for that whole life stage. It ignores contributions from all other locations. No exposure measurement errors are assessed in this approach. |

### 4.4.2 Supporting compatibility with older exposure studies

Many exposure early life exposure studies use the location at birth to assess the exposure of study members for the entire pregnancy or early life period. In order to make the result of the study comparable and compatible with other early life studies, the protocol will support a fourth exposure assessment method:

|  |
| --- |
| **Data Science-40**: The birth address assessment uses the birth addresses of study members to represent their locations for the entire early life exposure time frame (eg: conception until the last day of the first year of life). No exposure measurement error is assessed in this approach. |

### 4.5.3 Automatically compare results made by pairs of exposure assessment methods

Each of the exposure assessment methods generates average, median and cumulative exposures for each pollutant for each life stage of each study member. In order to save exposure scientists from doing tedious manual comparison, the protocol should be able to compare corresponding exposure values.

The exposure scientists decided that in any comparison of two methods, one method would be treated as the one that generated exact values and the other would be treated as the one that generated approximate values. The comparisons between pairs of methods would involve using percent error calculations:

|  |
| --- |
| **Data Science-41**: The protocol will compare pairs of exposure assessment methods and calculate percent error values for corresponding exposure results. |

### 4.5.4 Developing naming conventions for result files and result variables

The protocol produces over 400 variables, many of which have the same name but which are produced by different exposure assessment methods. In order to help researchers more easily understand the meaning of results, the protocol will adopt naming conventions that standardise the names and locations of result files. File names will comprise fragments that have standard meanings. For example, consider the naming convention used for following result file names:

res\_[early/later]\_mob\_cln\_exp\_[time stamp].csv

Here, “res” means results. The next part of the file name is either “early” or “later”, which describes whether the result is produced to support the early or later life analysis. “mob\_cln” means the results are for the cleaned mobility exposure assessment method. “exp” means exposure results.

|  |
| --- |
| **Data Science-42**: The protocol will produce CSV result files whose names follow standard naming conventions. These files will be generated within a directory that has predictable structure. |

Next, the ALGAE protocol will name variables using prefixes that help capture information about the context which produced them. For example, consider the two variables algae3108\_nox\_rd\_sum and algae3604\_nox\_rd\_sum. Both represent the cumulative exposure for nitrogen dioxide pollution coming from roads. However the differences in the prefixes of these two variables can help distinguish their contexts:

* algae3108\_nox\_rd\_sum means that the variable was produced using the cleaned mobility assessment method for the early life analysis.
* algae3604\_nox\_rd\_sum means that the variable was produced using the uncleaned mobility assessment method for the later life analysis.

|  |
| --- |
| **Data Science-43**: Variables that are produced by the protocol will have names which are prefixed by “algae[NNNN]\_”. The four digit number is designed to help uniquely identify the context which produced variables. |

Now that the naming conventions for both the file paths and variable names of result data sets, we are left with putting in the effort to help describe their meaning:

|  |
| --- |
| **Data Science-44**: The protocol will come with a comprehensive data dictionary that will define the meaning of result variables. |

**4. Software Engineering Decisions**



Testing and Performance

# 4 Software Engineering Decisions

This set of requirements covers mainly concerns about testing and performance. As we concluded in the discussion of business requirements, ALGAE has to be designed for testability. The protocol must be able to substitute real data for test data that can demonstrate that its algorithms are working correctly. The test cases we develop must be chosen judiciously because of limited developer resources. They must also operate automatically so that it is feasible to re-run large numbers of test cases in response to changes that may be made to the code. As well, we have to consider how efficiently the algorithm makes use of time and computer memory. Even if the protocol is able to run to completion, it can be difficult to estimate how well it will scale with large data sets. Its performance can be influenced by the computing resources that happen to be available during site visits to the cohort facilities. In the last part of this section, we discuss an example of design that shows the tension between wanting to better support testing and wanting to support better performance.

### 4.1 Supporting Testing

We divide the discussion about testing into three areas that we can design to reduce the scope of testing:

* constraining input data
* validating intermediate data sets
* designing test cases to test results.

#### 4.1.1 Constraining Input Data

One of the most effective ways of limiting the set of possible test scenarios we have to consider is to constrain the properties of input values wherever possible. We can support this task in four ways:

* provide clear, concise documentation to help guide cohort projects in transforming their cohort-specific data into the input tables that are expected in the protocol
* prevent errors by standardising the way input values are represented
* identify empty required field errors and fail as soon as possible
* identify duplicate field errors and fail as soon as possible

The first and most important way we can help minimise variability of input data is through end user training. Providing the software alone merely enables other projects to use it. However, without clear documentation, they may choose to use data that the protocol does not expect.

|  |
| --- |
| **Software Engineering-1**: The protocol will come with clear, concise documentation that project staff can use to transform their data into the input tables that are expected by the software. |

Next, we must address the potential problems that come with the variance of input data we support for them. The business perspective favours having the protocol accommodate a wide variety of ways that researchers may represent the data that are destined to form the input tables. By allowing cohort projects to have the flexibility to represent yes/no field values and empty field values, the protocol may make it easier for them to start using it.

However, the software engineering perspective favours standardising input values to simplify the code, make it less error-prone, and minimise the need for repetitive test scenarios that may only differ by the way the these values are represented. We don’t want to put in extra branches in the code to cope with variations of a “yes” value: “Y”, “y”, “yes”, “TRUE”, “true”, “T”, “t”, and 1. Adding extra code for handling more diverse inputs makes the software more error prone and more costly to maintain. To safeguard against encountering errors, we would then need to add test scenarios to consider each of these representations. The same case goes for values of “No” and for representations of empty field values.

The data science perspective favours preserving the provenance of original data so that analysts can trace results back to the input values which helped form them. Therefore, in order to reconcile the need to allow variation in input values provided by users and standardisation in input values for most of the algorithms, we will use staging tables:

|  |
| --- |
| **Software Engineering-2**: The protocol will create a staging table for each table of original input data. The staging table will use a standardised way for representing yes/no values and empty field values. The rest of the protocol will operate on the staging tables and not the input tables. |

There are many fields whose values need to be standardised. Rather than trying to repeatedly write similar pieces of code to standardise different table fields, the operations should be centralised in reusable database functions. The functions can be rigorously tested independently of the input data sets. Once they are shown to work, testing can assume that the protocol will standardise values correctly in every place where these functions are called.

|  |
| --- |
| **Software Engineering-3**: Consolidate code for standardising input fields within well-tested database functions. Later on in testing, assume standardisation functions will work and then rely on just using standard forms to express input data for test cases. |

Now that we have identified any blank fields, we should determine if any of them should be replaced with imputed values. Decisions for handling missing values rest with the scientific use case. For example, in the use case study, Imperial College epidemiologists decided that if a study member had a blank value for gestation age at birth, the value should be imputed with a typical value.

From the software engineering perspective, imputing blank values for a field simplifies code that that would be used to process it. In the case of the gestation age field, code that uses it in life stage calculations does not need to check if it’s empty.

|  |
| --- |
| **Software Engineering-4**: When it is appropriate for the scientific use case, impute blank values. For example, blank gestation age values will be imputed with a default value for gestation age in weeks. Imputing field values helps simplify code that processes them. |

Another strategy for dealing with errors is to consider ignoring them. For example, consider the field absent\_during\_exp\_period, which is a field in the study member data table. This is a data quality field that is provided by the cohort, but which is not used in any critical operations by ALGAE. If this field is left blank, none of ALGAE’s operations would be affected. Ignoring blank field values here is an option, but if cohort researchers felt they needed them to be filled for some sensitivity study, then they would want to be alerted as early as possible that they had to fix their input data.

|  |
| --- |
| **Software Engineering-5**: Some missing value errors will be important to users but will have no effect on the protocol’s algorithms (eg: absent\_during\_exp\_period and at\_1st\_addr\_conception). Reject rather than ignore these errors in order to help users fix their input data as soon as possible |

When an error cannot be prevented, fixed or ignored, the remaining option is to reject it. It is a common approach in software engineering to design systems that fail fast: if a program is destined to fail because of an error, it should do so as soon as possible. Considering that the protocol would be run off-site in short, expensive windows of development, it makes sense to use this approach to prevent hours of computing to be wasted.

|  |
| --- |
| **Software Engineering-6**: Design the protocol to fail fast: if it is going to fail because of an error, then make it fail as soon as possible. |

We can apply this approach to the two types of input errors we will consider: duplicate field errors and blank required field errors. Once the staging tables have been created, we can leverage the database’s own features to identify both kinds. To identify duplicate field errors, we can attempt to make a field a primary key:

ALTER TABLE study\_member\_data ADD PRIMARY KEY (person\_id);

If this operation fails, we know that at least one of the values for person\_id is repeated at least once in the study\_member\_data table. By identifying duplicate key values now, we can prevent database join errors that may cause more rows to be generated when tables are linked together.

|  |
| --- |
| **Software Engineering-7**: Identify duplicate key errors by applying a database constraint which tries to use a field as a primary key. |

Similarly, we can identify empty field values by attempting to apply constraints that do not allow blank values:

ALTER TABLE staging\_geocode\_data ALTER COLUMN geocode SET NOT NULL;

|  |
| --- |
| **Software Engineering-8**: Identify required field values by applying a database constraint which tries to forbid a field from having empty field values. |

Again, if this constraint fails, the users can be informed that they have mistakenly left a critical field value blank. In both of these cases, validation fails before the rest of the protocol attempts to use the data. This technique is used again when we validate intermediate tables.

### 4.1.2 Validating Intermediate Data Sets

Now that we have constrained the input data as much as possible, we deal with the next potential source of errors: the intermediate data sets produced by the protocol. Here we rely on aspects of validation, auditing and database techniques that can help minimise the risk of losing or gaining more results than we expect.

Given the large data sets that are expected to be processed, it is tempting to begin optimising the performance queries early in the development process. Often, it is desirable from a performance perspective to combine a sequence of discrete query operations into a single monolithic query that can be run entirely within the computer’s memory. Most of the performance gains happen because it is faster for the database to read and write data from RAM than it is to read and write data from disk. How well this approach works can depend on the database’s query optimisation features and the availability of memory. When the database is no longer able to run a query entirely within memory, its performance will degrade as it begins to use more swap space.

There are two main drawbacks of favouring the use of complex monolithic queries over a sequence of simpler queries. First, programmers may tend to make more coding mistakes when they construct large complex pieces of code. Second, can be difficult to trace the source of errors from results back to input data.

Writing complex monolithic queries as a sequence of simpler ones often makes it easier for analysts to identify when an error is created by the program. The approach also allows tests to be developed which can identify errors that may be masked in the final result tables. In a protocol that is characterised by a long sequence of steps, there is a risk that errors will be less detectable. This is especially the case in ALGAE, where errors may be hidden in aggregated results.

The need for the database to read and write temporary tables that may contain mostly the same data can introduce a significant performance drawback. An important design question to ask is: “Should we favour developing features that minimise errors and improve testing over features that can help the program perform better?”

In the use case study, we decided that the answer to this question was “yes”, because the cost of a returning to Bristol to correct errors was too high. Therefore:

|  |
| --- |
| **Software Engineering-9**: It is more important that the program performs correctly than it is that the program performs quickly. |

The decision makes it easier to favour using temporary tables:

|  |
| --- |
| **Software Engineering-10**: Favour expressing complex operations through a sequence of simple temporary tables rather than by using a single, complex monolithic query to do the same work. |

In activities that rely on a sequence of temporary tables, we have to be wary of errors that may cause an unexpected loss or gain of records. This is especially important in the way we have supported sensitivity variables that are supposed to be provided for all study members. For example, consider that in the first temporary table for sensitivity variables, we indicate whether gestation age has been imputed for each study member. At a later step, we try to identify members who have a bad geocode within the address periods that cover their exposure time frame. We want to make sure that when we report the final sensitivity variable table, that all study members from the first table are included, not just the ones who showed a bad geocode. Here we want to ensure that queries used to identify characteristics of some study members are used to provide values for all rather than filter values for some.

A common database technique for preserving a consistent number of rows in temporary tables is to use LEFT JOIN operations. Consider two tables we want to link together: a main table T1 that contains study member information we’d like to preserve in final results and an auxiliary table T2 that identifies an interesting property that may only apply to a subset of study members. If we use T1 left join T2 using study member ids, then the result will preserve all the study members in T1 and add data from T2 if information is available. If a study member id appears in T2 but not T1, then the resulting table will indicate an empty property value for that person. However, some value for the property will exist for all the study members that were included in T1.

|  |
| --- |
| **Software Engineering-11**: Where it is important to preserve rows from one temporary table to the next, rely on LEFT JOIN operations to link the data applicable to all study members with the new data that may only be applicable to some. |

Another way to minimise the risk of having an unexpected number of rows generated in temporary tables is to compare the keys that appear in the first and last tables. For example, we use a sequence of seven tables to build up sensitivity variables for each study member. As a check to make sure the study member ids have been preserved throughout the activity, we compare will compare the ids found in the first and last table:

|  |
| --- |
| **Software Engineering-12**: Consider a sequence of successive temporary tables marked by T1…Tn that are meant to build up a growing collection of field values describing the way cohort data have been processed for each study member. The protocol will compare the keys between T1 and Tn so that they have the same number and that all the keys in one are found in the other. |

In order to make it easier for analysts to identify the source of errors and to prevent them from being masked by results, we tend to rely on using temporary tables to refine information that will eventually become part of results. In order to reduce the potential for error by using more tables, we try to use LEFT JOIN statements to preserve table keys and then compare the keys of first and last temporary tables to ensure there has been no change.

### 4.1.3 Designing Test Cases to Test Results

So far we have constrained input values and relied on using successions of temporary tables that can be used to audit intermediate calculations and help identify the source of errors. We’ve also recommended using techniques which minimise the risk of losing data or gaining duplicates through incorrectly linking tables together. We are left with testing the results.

In order to make test cases practical to maintain and apply, the protocol needs a way of supporting automated testing so that when changes are made to the code, the tests can be re-run.

|  |
| --- |
| **Software Engineering-13**: The protocol will support a means of automatically comparing the actual results generated by the program with expected results that are derived through manual calculation. |

Next, we need to consider the design of an individual test case. Its purpose should reflect a requirement of the system, its implementation should reflect efficient coverage of execution pathways in the code and its results should minimise the risk that one error could be masked by others. We made the following decisions, both of which are part of popular testing approaches:

|  |
| --- |
| **Software Engineering-14**: Each test case should only test one feature. |

|  |
| --- |
| **Software Engineering-15**: Within a test suite, test data related to each study member should remain the same, except for data that relate to the features being tested. |

Next, we need to support documenting test cases so that it is easier to understand what they do when expected results are compared with actual results. Test data for each test case are scattered across four input files and it can be difficult to piece them together. We can make it easier to understand test data if we use the names of test scenarios for study member identifiers. For example, in the test suite that examines the way address periods are adjusted, we use “missing\_start\_date” as a study member who has an address period that is missing a start date.

|  |
| --- |
| **Software Engineering-16**: In order to make test data easier to understand use the names of test scenarios as study member identifiers. |

Second, allow each input data set to have a comment field. It will make it easier to annotate test data and to link test descriptions with the result of comparing actual versus expected results.

|  |
| --- |
| **Software Engineering-17**: In order to make test data easier to understand use the names of test scenarios as study member identifiers. |

We now turn to decisions about how to craft the test data. The expected values for test cases have to be simple enough to be easy to calculate manually. The emphasis of test case design is more about addressing variance in the data sets rather than the frequency of particular values or the volume or records in them.

For example, we do not need thousands of different addresses to simulate real cohort data. We need only a few to allow some study members to move from one location to another. We do not need realistic birth dates to fit the use case cohort data, but one of them should cause a study member’s life stages to span the extra day in a leap year. There is no testing benefit for having the earliest and latest born members in a test suite far apart. Although that may be realistic, from a testing perspective, it just means that an unnecessary amount of daily exposure records for each location will be generated.

Our choice of field values does not have to be complicated either. ALGAE only cares that an exposure value is a number. In the use case study, exposures are expressed using numbers with many digits of precision. However in testing, using similar numbers would increase the risk of making errors in manual calculations.

|  |
| --- |
| **Software Engineering-18**: Each test case should use the minimal amount of input data it needs to exercise a feature. The values should be simple enough to make them amenable to being used in the manual calculations that produce expected test results. |

Now we turn simplifying data used to support testing exposure calculations. It is the most complicated part of testing and we found that manual calculations that used daily exposure values were error prone. For some test cases, we need enough variation in aggregated exposures to help clearly distinguish average and median values. We can support variation by using some combination of the following ways:

* make the exposure value for one pollutant very different than the exposure value for another pollutant
* adjust the duration that a test study member stays at one geocode
* at a given geocode, adjust the exposure levels so they vary over time
* make exposure at one geocode different from another

The way we decided to simplify test exposure data is shown in the table below. Exposures for each pollutant at each geocode were kept constant. When variation was needed to distinguish average and median values in life stage aggregations, we adjusted the residential address histories to influence the amount of time test members stayed at geocodes that had different exposure levels. Simple sequences were used to ensure that differences between successive geocodes (going down) and between successive pollutants (going to the right) were unique. The values generated through the sequences made it easier to trace the source of error for both aggregated exposure values and aggregated exposure misclassification values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Geocode** | **PM10\_rd** | **NOX\_rd** | **PM10\_gr** | **NAME** | **PM10\_tot** |
| a1 | 1 | 2 | 3 | 4 | 5 |
| a2 | 3 | 4 | 5 | 6 | 7 |
| a3 | 6 | 7 | 8 | 9 | 10 |
| a4 | 10 | 11 | 12 | 13 | 14 |
| a5 | 15 | 16 | 17 | 18 | 19 |
| a6 | 21 | 22 | 23 | 24 | 25 |
| a7 | 28 | 29 | 30 | 31 | 32 |

The main features of our approach for generating test exposure data are summarised through the following decisions:

|  |
| --- |
| **Software Engineering-19**: For each pollutant at each geocode, assign stepped pollution values that are constant over time but which vary significantly between geocodes and between pollutants. |

|  |
| --- |
| **Software Engineering-20**: For test cases that need to demonstrate variation between average and median life stage exposures, induce differences by adjusting the boundaries of address periods spent in different locations. |

|  |
| --- |
| **Software Engineering-21**: Design stepped exposure generation functions so that the differences between successive geocodes and the differences between successive pollutants is unique. |

So far in our discussion about testing, we have constrained the allowable values for input data, we have taken measures to prevent data linking errors and we have covered providing protocol facilities that can support automated testing activities. We have invested care in designing test data sets which are complex enough to exercise features but which are simple enough for results to be calculated by hand. We can now apply our last way of making testing more effective by deciding what feature areas we can ignore.

ALGAE treats all of the pollutant values in exactly the same way. The design of the test data will ensure that exposure results are guaranteed to be different for each of them. If they are the same, then it would suggest that there is some copy-and-paste error in the code where a field name has not been changed. This kind of verification can be done by eye. If the values are all different, then we will make the assumption that test cases can focus on using one pollutant.

|  |
| --- |
| **Software Engineering-22**: Testing will assume that ALGAE treats all pollutants in the same manner. Testing multiple pollutants will be limited to inspecting by eye that aggregated exposure values are in fact different from pollutant to pollutant. |

|  |
| --- |
| **Software Engineering-23**: Limit the scope of testing by focusing test cases on only one pollutant. |

## 4.2 Supporting Performance Concerns

It is difficult to make accurate performance estimates for how long the protocol would take to process its data sets. Even assuming that the protocol encountered no errors which caused it to fail, the time it would need to produce results would depend on two factors:

* the computing resources made available by the cohort facilities to run the protocol
* the other processes might be competing for these same computing resources.
* the way the database optimises queries

### 4.2.1 Save Time by Failing Fast

In this section, we discuss way that will likely improve efficiency despite these three factors. We have already discussed the one way: if the protocol is going to fail, it should fail as quickly as possible. In practice, this leads to validating data as early as possible, and throwing exceptions in the code as soon as an error is detected. This does not improve the efficiency of successful runs, but it shortens the time needed by unsuccessful ones.

The remaining ways are meant to help speed up the code, whether or not they are part of successful program runs.

### 4.2.2 Favour using Create Table As Select (CTAS) statements over either update or insert operations affecting many records

The protocol design already supports this preference. Our concern about preserving the state of original input data sets means that we would copy changes in newly created staging tables rather than try to update changes in existing input fields.

Our interest in creating an audit trail of changes to facilitate testing also compels us to use CTAS statements. Cleaning address periods is done using a succession of temporary tables, where modifications to fields from one table are stored in fields created in the next. An example of this is shown in how we maintain multiple versions of the start and end dates in address periods as separate fields. The final version of the table containing address periods contains the fields: start\_date, adjusted\_start\_date and fin\_adjusted\_start\_date. Each field represents the value of start date at different points in the cleaning process.

Update statements are useful when a few changes are made in a large data set. However, it is difficult to know whether input data files will need to be cleaned a lot or a little. Therefore, it is more likely in general that CTAS tables will prove a better option.

Another reason why it can be desirable to avoid using UPDATE, INSERT or DELETE statements is that the performance of these operations can be degraded by indexing.

### 4.2.3 Apply Indexes where Appropriate

Adding indices improves the speed with which the database can find values in columns of a table. As a general rule of thumb, indices work well with fields that appear in GROUP BY, JOIN and WHERE clauses of query statements. However, the topic of optimising queries through indexing is complicated. For example, the value of indexes can be influenced by the frequency of field values. For example, if a field value “X” appears once every thousand rows of a column in a large table, the index will usually be used. However, if “X” appears in most values for the column, the database may choose to do a full table scan, where it tries to scan each row in the table and check for the validity of the match.

Improving performance is an area of future development in ALGAE. Unfortunately, most of the project time was spent making it work, and then making sure it was working correctly.

## 4.3 Sharing Code: Good Idea or Bad Idea?

In order to accommodate the use of daily exposure records in the early life analysis and yearly average exposure records in the later life analysis, different code was developed for each analysis. However, as we tried to add analogous table fields for each analysis, the code began to become more error prone. Differences in the way values were being aggregated meant that more testing effort would be needed to ensure that different ways of doing the same aggregation were each generating correct values.

In the early life analysis, SUM(…), AVG(…) and median(…) operations were easy to do because they were all given a collection of column values for some pollutant. In the later life analysis, the sum was calculated by multiplying the annual average value for the number of days that a person spent at a given location for a given calendar year. The average was calculated by multiplying the annual exposure value by a fraction where the numerator was the number of days spent at a location for a calendar year and where the denominator was the number of days in that calendar year.

Calculating median exposures for the later life analysis proved more complicated. PostgreSQL currently does not come with a built-in function for calculating the median for a set of column values. Instead, we had to adapt an open source solution which accepted an array of numeric values as input. It was not clear whether the function resulted in expensive computation or not, but we knew it required an array of column values.

We initially used the number of days spent at each location for each calendar year to help generate an array of values that the median function could then order to find the middle. However, these calculations resulted in errors. We later saw more potential errors in the sum and average calculations. The errors were part of code that was designed to replicate the effect of daily exposure aggregations, but using annual values. The mistakes formed part of the motivation to make a return visit to Bristol. Given the expense of fixing errors through long repeated runs, we gave thought to redesigning this part of the code base.

We decided that it would be far simpler to process if, in the later life analysis, annual exposure values were used to generate daily exposure values. The rest of the program could then use the same code in both early and later life analysis. By making both analyses share the same code for aggregating results, we could be certain that if test cases for exposure assessment worked for the early life analysis, they would work for the later life analysis as well.

The decision to share code to reduce the potential for errors came at the cost of generating a large amount of daily exposure records. Whereas before, the exposure data set would contain daily exposure records for an overall exposure time frame of about two years, the new way of running the protocol would require daily exposure records at all geocodes for fifteen years.

We were not sure how long it would take the new protocol to run, but it turned out that the later life analysis took well over 40 hours to run. However, two aspects of the work helped make this result less alarming:

* rigorous testing on simplified exposure calculations made us confident that the protocol would process the input values to completion
* much of the work was set to run overnight, thereby using computing time that would normally have been wasted anyway.

The design question which remains – was it worth making the analyses share code even though it could drastically increase the time needed for one of them to complete? We feel it was worth the effort, simply to guarantee that a return visit would not be necessary to make corrections. However, we were surprised to discover how long the task took and now realise that performance tuning for such large data sets is an area of future development.

## 4.3 Image References

<https://commons.wikimedia.org/wiki/File:Wrench.svg>

<https://openclipart.org/detail/202775/gear-3d>

<https://openclipart.org/detail/179428/clipboard>

<https://openclipart.org/detail/99937/speedometer>

<https://openclipart.org/detail/210403/misc-plain-metal>

<https://openclipart.org/detail/19182/gold>

<https://openclipart.org/detail/30961/briefcase>

<https://openclipart.org/detail/204068/bpm-time-symbol>

<https://commons.wikimedia.org/wiki/File:Pound_Sign.svg>

<https://openclipart.org/detail/175194/padlock>

<https://openclipart.org/detail/business-people-siluete---by-systemedic>

<https://openclipart.org/detail/203314/science>

<https://commons.wikimedia.org/wiki/File:Oxygen480-actions-office-chart-line-stacked.svg>

<https://commons.wikimedia.org/wiki/File:Oxygen480-actions-office-chart-line-stacked.svg>

<https://commons.wikimedia.org/wiki/File:Stethoscope_(PSF).png>